2019

DEEP LEARNING FOR IMAGE RESTORATION AND ROBOTIC VISION

Yixin Du
West Virginia University, yidu@mix.wvu.edu

DEEP LEARNING FOR IMAGE RESTORATION AND ROBOTIC VISION

Yixin Du

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Xin Li, Ph.D., Committee Chairperson
Donald A. Adjeroh, Ph.D.
Yanfang Ye, Ph.D.
Victor Fragoso, Ph.D.
Yu Gu, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2019

Keywords: Deep Residual Learning, Image Super-Resolution, Autonomous Robots

ABSTRACT

Deep Learning for Image Restoration and Robotic Vision

Yixin Du

Traditional model-based approach requires the formulation of mathematical model, and the model often has limited performance. The quality of an image may degrade due to a variety of reasons: It could be the context of scene is affected by weather conditions such as haze, rain, and snow; It's also possible that there is some noise generated during image processing/transmission (*e.g.*, artifacts generated during compression.). The goal of image restoration is to restore the image back to desirable quality both subjectively and objectively. Agricultural robotics is gaining interest these days since most agricultural works are lengthy and repetitive. Computer vision is crucial to robots especially the autonomous ones. However, it is challenging to have a precise mathematical model to describe the aforementioned problems. Compared with traditional approach, learning-based approach has an edge since it does not require any model to describe the problem. Moreover, learning-based approach now has the best-in-class performance on most of the vision problems such as image dehazing, super-resolution, and image recognition.

In this dissertation, we address the problem of image restoration and robotic vision with deep learning. These two problems are highly related with each other from a unique network architecture perspective: It is essential to select appropriate networks when dealing with different problems. Specifically, we solve the problems of single image dehazing, High Efficiency Video Coding (HEVC) loop filtering and super-resolution, and computer vision for an autonomous robot. Our technical contributions are threefold: First, we propose to reformulate haze as a signal-dependent noise which allows us to uncover it by learning a structural residual. Based on our novel reformulation, we solve dehazing with recursive deep residual network and generative adversarial network which emphasizes on objective and perceptual quality, respectively. Second, we replace traditional filters in HEVC with a Convolutional Neural Network (CNN) filter. We show that our CNN filter could achieve 7% BD-rate saving when compared with traditional filters such as bilateral and deblocking filter. We also propose to incorporate a multi-scale CNN super-resolution module into HEVC. Such post-processing module could improve visual quality under extremely low bandwidth. Third, a transfer learning technique is implemented to support vision and autonomous decision making of a precision pollination robot. Good experimental results are reported with real-world data.

Acknowledgments

Without the talent of being a swift horse, it is my greatest honor to meet a good judge of talent: Professor Li, Xin at West Virginia University. His spirit of play motivated me to try new things. His insight and professionalism set a perfect example that I want to achieve in my career. His guidance helps me gain enough self-confidence to meet any new challenges in the future. The value of the Ph.D. study experience under Prof. Li's advisement is timeless in my life.

I'm so grateful to my own creator for supporting me to study abroad for nearly eight years. I also appreciate the encouragement from my wife during those dark periods. Without their support, this dissertation could not be accomplished.

I would like to offer my special thanks to Dr. Victor Fragoso, Dr. Yanfang Ye, and Dr. Donald A. Adjeroh and Dr. Yu Gu for their valuable and constructive suggestions during the planning of this research work. Their willingness to dedicate their time so generously has been very much appreciated.

I would like to express my very great appreciation to Dr. Yu Gu for giving me the opportunity to collaborate in the robotic project. I wish to acknowledge the help provided by Mr. Jared Strader as well as many other team members.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The surge of Artificial Intelligence (AI) in the last decade has brought revolutionary changes in nearly every corner of industry and our daily life. Many AI milestones, which experts expected as decades away, has been reached in the last five years. Thanks to the advance in computing hardware such as powerful CPU and GPU, modern AI technology is capable of dealing with large-scale data and problems. Computer vision, one of the most studied area in computer science, benefits from the development in AI really well. Recently, AI-based approaches, specifically deep learning-based approaches, have dominated in most of the computer vision problems. Regardless of whether it's high level vision task such as scene understanding, or object detection/recognition (middle level vision task), or image processing (low level vision task), deep learning has proved to have superior performance when compared with traditional approaches. Thus, this dissertation explores to implement learning-based approaches in image restoration and robotic vision based on image classification.

Many techniques have been proposed in deep learning to achieve better performance. We give an overview of deep learning from two perspective: the type and the size of network. From the type perspective, direct non-linear mapping, residual learning, and generative adversarial network are three millstones in the field. Direct non-linear map-

ping achieved good performance in 2012-2014 such as SRCNN [18]. But it was replaced by residual learning [33] due to the vanishing gradient problem [5]. Vanishing gradient problem happens when more layers are used, the gradient of the loss function approaches zero, making the network hard to train. Other than distortions, visual quality has draw researchers' attention, thus, generative adversarial network has been proposed for perceptual optimization such as SRGAN [52]. From the size perspective, learning-based approaches in early stages only have a few layers such as FSRCNN [20] which has eight layers, the model is only 51 kilobytes. Most recent neural network could have hundreds of layers such as RCAN [112] which has 400 layers, and the model is 59 megabytes.

Picking an appropriate network, in terms of network type and size, is very important for different problems. In **Image Dehazing,** we favor residual learning instead of direct non-linear mapping to avoid the vanishing gradient issue. Meanwhile, the perception-distortion trade off also deserves deliberation because sometimes the visual quality is more favorable compared with distortion performance measure such as pixel-to-noise-ratio (PSNR). In **Loop Filtering and Image Super-Resolution for Video Coding,** we solve recourse-constrained problems with limited bandwidth, thus, the deeper the network does not mean the better. In **Agricultural Robotics,** transfer learning from one task to another is a smart choice if the network body has rich features. This could avoid the lengthy training-from-scratch alternative. This dissertation explores two different areas (image restoration and robotic vision based on image classification), but with one unifying theme, *i.e.*, picking the right neural network for different problems.

Image Dehazing. An image taken from outdoor often suffers from low contrast and degraded color due to haze, fog, mist, etc. How to restore the visual quality of hazy images is called "image dehazing" or "haze removal". Traditional approach aims at finding the so-called transmission map in order to recover the haze-free image in a two-step manner. In this paper, we propose to reformulate and solve dehazing bypassing transmission map estimation, fulfill the dehazing task as an end-to-end optimization. We use recursive deep

residual network to improve the objective quality of dehazed image, and we further improve the perceptual visual quality via generative adversarial network. They are presented in Chapter 2.

Loop Filtering and Image Super-Resolution for Video Coding. Loop filtering aims at removing compression artifacts and improving visual quality of a video. It is an important component in High Efficiency Video Coding (HEVC). Traditional loop filters in HEVC such as deblocking filter has limited performance with respect to filtering accuracy. Inspired by the success of deep learning, some Convolutional Neural Network based loop filters have been proposed to address the issue of traditional filters. Follow this line of thought, we propose a neural network filter for further coding efficiency improvement. When compared with traditional filters, the proposed approach could fulfill multiple filtering tasks with only one single model. It is also better when compared with current CNN-based loop filters with respect to BD-rate savings. Section 3.1 will review previous loop filters including both traditional model-based and CNN-based approaches, propose our new model, and report experimental results. We also present a content-weighted idea which has been filed as a pattern in Section 3.2. Another topic we discover is building a CNN-based super-resolution module to improve visual quality in HEVC. There exists the following scenario: when bandwidth is really limited, the compression uses very large Quantization Parameter (QP), as a result, the quality of reconstruction is very low. An alternative approach to this problem is to use super-resolution in HEVC. One could down-sample the video at the beginning, use a smaller QP to compress the video, and finally do an up-sampling. The counter-part is the alternative approach will produce smaller compression loss and down sample loss, the original approach will have larger compression loss. So eventually we are comparing which loss is smaller between the two. We argue that incorporating a CNN super-resolution module (the alternative approach) is better than compressing the video under original resolution. The methods and experimental results are shown in Section 3.3.

Agricultural Robotics. Due to the rapidly increasing of human population and the rigorous demand of high quality food, it is essential to figure out new ways to aid agriculture. Productivity has to be increased to meet the call, while human labors are becoming more and more expensive. Most of agricultural tasks involve repetitive and lengthy which makes robotics a great fit to address the aforementioned problem. Thus, agricultural robotics are rapidly gaining interest in many existing fields. The decline of natural pollinators is one of the critical issues faced by the agricultural sector today [72]. To resolve this issue, robotic precision pollination technique has been proposed by a group of researchers at West Virginia University (WVU). A fully autonomous precision pollination robot is designed for performing pollination toward bramble plants in a greenhouse. I was responsible for the computer vision subsystem of the robot, and Chapter 4 will mainly discuss my contribution in robotic vision of "BrambleBee".

The contribution of this dissertation can be summarized as follows: First, we propose to reformulate haze as a signal-dependent noise which allows us to uncover it by learning a structural residual. Based on our novel reformulation, we solve dehazing with the state-of-the-art recursive deep residual network and generative adversarial network which emphasizes on objective and perceptual quality, respectively. Second, we replace traditional filters in HEVC with a Convolutional Neural Network (CNN) filter. We show that our CNN filter could achieve 7% BD-rate saving when compared with traditional filters such as bilateral and deblocking filter. We also propose to incorporate a multi-scale CNN super-resolution module into HEVC. Such post-processing module could improve visual quality under extremely low bandwidth. Third, a transfer learning technique is implemented to support vision and autonomous decision making of a precision pollination robot. Good experimental results are reported with real-world data.

Chapter 2

Single Image Dehazing with Deep Learning

2.1 Introduction

The World Meteorological Organization (WMO) classifies environmental obscuration into multiple categories as shown in Figure 2.1, such as fog, mist, haze, smoke, and dust. In this work, we do not explicitly distinguish these environmental obscuration, we generally describe them as haze since it is the most frequently used word in computer vision under the domain of haze removal. Images taken under the aforementioned weather conditions often has low contrast and degraded color. As a result, objects tend to occluded especially at far distance from camera, and the image itself looks unpleasing. How to restore the visual quality and make objects more easier to see is called "single image dehazing" in literature.

Removing haze from image is an ill-posed problem for the following reasons: First, in image denoising, the noise is often times consistent in the image and it can be described with a single σ if the noise is Gaussian. Unlike noisy image, hazy image has non-uniform haze appearance, i.e., the longer the distance from camera, the heavier the haze. This

(a) Fog

(b) Mist

(c) Haze

(d) Smoke

(e) Dust

Figure 2.1: Environmental obscuration defined by the World Meteorological Organization (WMO).

brings challenges as dehazing has to take scene depth into consideration. Second, previous approaches utilize an image formation model which connects observed hazy image with scene radiance (unknown target), transmission map, and atmospheric light. Based on this formation model, the problem of single image dehazing boils down to estimating the transmission map and atmospheric light. However, to the best of our knowledge, this formation model only provides an estimation of real hazy condition, how accurate this model is remains unknown. Last but not least, the variation in global atmospheric light (e.g., nighttime vs. daytime) makes dehazing even more challenging.

Previous approach heavily relies on the haze formation model and the estimation of transmission map and atmospheric light. In this work, we challenge the conventional wisdom by proposing a novel reformulation. Our reformulation models haze as a signal-dependent noise, thus, it allows us to uncover haze as structural residual. Based on the reformulation, we propose a recursive deep residual network and a generative adversarial network which aims at objective and subjective dehazed image quality, respectively. Ex-

tensive experimental results show that our approach outperforms other competing methods in terms of both subjective and objective visual quality of dehazed images.

The rest of this Chapter is arranged as follows: we present related work on single image dehazing, Deep Residual Network (DRN), and Generative Adversarial Network (GAN) respectively. Then, we show the haze formation model as well as our reformulation. We also present the architecture of our DRN and GAN, along with the experimental results. Finally, we conclude this Chapter.

2.2 Related Work

2.2.1 Related Work on Dehazing

Model-based approaches toward single image haze removal are often based on some haze-relevant priors. In [31], a Dark Channel Prior (DCP) was proposed based on the observations that in a haze-free image, any local patch would contain some pixels of low intensity values in at least one color channel. The thickness of the haze is then directly estimated using the DCP and a haze-free image is recovered. The advantage of this approach is it's effectiveness toward haze removal, but only limited to a certain type of images. Under the condition when an image contains a lot of sky pixels, the model has limited performance. Besides, it is computationally extensive which makes real-time dehazing almost impossible. In [70], a factorial Markov Random Field model was adopted to jointly estimate the scene albedo and depth. A Bayesian probabilistic method is introduced to estimate scene albedo and depth using their latent statistical structures. The natural image and depth statistics are treated as priors of hidden layers. A canonical expectation maximization algorithm is proposed to estimate the depth of a scene. The authors reported accurate factorization on challenging scenes of the proposed method. The main disadvantage is that it tends to produce over-saturated images. In [65], the inherent boundary constraint of the transmission function was exploited during the estimation, which has been called contextual regulariza-

tion. The constraint is combined with a weighted contextual regularization, forming an optimization problem to estimate the unknown scene transmission. The authors also presented an efficient algorithm to solve the problem based on splitting the variables. The main advantage of this method is that it requires only a few general assumptions to restore faithful and fine image details. But the shortcoming is that it is also computationally intensive. In order to improve the efficiency, many other filters have been proposed, such as standard median filtering [26], median of median filter [94] , and guided image filter [30]. In non-local image dehazing [6], distance map and haze-free images are jointly estimated from haze-lines characterizing the linear-spreading structure of pixels within a given cluster in the RGB space. The work is based on the key idea that pixels in a given cluster are often non-local, spread over the entire image plane and located at different scene depth. These varying scene depth is translated into different scene depth. The author has characterized the color cluster as a line in RGB space to recover the distance map and haze-free image. One disadvantage is that it assumes a fixed distribution of 3D lines from the air light which limits it's power to describe an actual scene [7]. All those dehazing methods estimate the scene depth or transmission map to facilitate model-based image restoration.

Inspired by the success of deep learning in various low-level vision tasks such as image super-resolution [18, 19] and image denoising [100, 111], learning-based approaches have also been proposed for singe image dehazing in recent years [79, 11, 54]. The common foundation behind those works is the creation of training data - a large number of synthetic hazy images based on ground truth depth information. In [79], a multi-scale Convolutional Neural Network (MSCNN) was proposed to estimate transmission map from an input of hazy image. It consists of a coarse-scale net which aims at globally predicting the holistic transmission map of a scene, and a fine-scaled network to further refine the transmission map estimation locally. The authors have adopted a synthetic dataset of hazy images and their corresponding transmission maps. They have reported favorable performance on both synthetic and real-world images with respect to quality and speed. While

one of the limitations is that even though it has adopted the learning based approach, but it is only confined to the transmission map estimation part. The air light estimation and scene radiance recovery are still similar as traditional model based approach which basically makes it a two-step dehazing approach. In [11], authors developed an end-to-end dehazing network called DehazeNet in which network layers are specially tailored to fit assumptions/priors in the scenario of image dehazing. It takes a hazy image as input, and outputs its medium transmission map that is then used to recover the haze-free image via atmospheric scattering model. Haze-relevant features are extracted from the layers of max-out units. A nonlinear activation function called Bilateral Rectified Linear Unit is proposed to improve the quality of dehazed image. Similar as MSCNN, DehazeNet is also an indirect learning based dehazing approach which aims at transmission map recovery. The rest haze-free image recovery is still model-based. Most recently, the so-called All-in-One Dehazing (AOD)[54] further develops this line of idea by unifying the estimation of transmission map and global atmospheric light into one module called K-estimation leading to the current state-of-the-art performance in this area. AOD directly generates the clean image through a end-to-end CNN. This design makes it very easy to concatenate AOD with other middle/high level computer vision tasks' networks. The author not only showed the visual qualify of dehazed image, they also experimentally pointed out the effectiveness of dehazing on object detection and recognition. However, AOD requires large amount of training data due to the complicated network design, whereas our DRL requires way less training image to achieve good performance. Densely connected pyramid dehazing network (DCPDN) is proposed in [110] to learn the transmission map and atmospheric light together. It has good performance in terms of distortion but limited visual quality. Gated fusion network (GFN) [80] and conditional GAN (cGAN) [57] are also proposed to address single image dehazing. Table 2.1 summarizes the contribution and limitation of some of the state-of-the-art approaches.

Table 2.1: Comparison of previous works on image dehazing.

	Paper	Year	Contribution	Limitation
Model -based	He *et al*. [31]	2011	Dark Channel Prior	Limited performance on sky pixels.
	Nishino *et al*. [70]	2012	Bayesian model	Overly saturate images.
	Meng *et al*. [65]	2013	Contextual regularization	Computationally intensive.
	Berman *et al*. [6]	2016	Haze lines	Fixed distribution of 3D lines.
Learning -based	Ren *et al*. [79]	2016	Multi-scale CNN	Only applicable to transmission map estimation.
	Cai *et al*. [11]	2016	Bilateral Rectified Linear Unit	Only applicable to transmission map estimation.
	Li *et al*. [54]	2017	All-in-one architecture	Large amount of training data needed.
	Zhang *et al*. [110]	2018	Pyramid network	Visual quality is limited.
	Ren *et al*. [80]	2018	Gated fusion network	Confidence map is needed.
	Li *et al*. [57]	2018	Conditional GAN	Large network size.

2.2.2 Related work on Deep Residual Learning

With the availability of large-scale dataset and advances in deep learning methods, the Convolutional Neural Networks have shown remarkable success in dealing with various computer vision tasks. Simonyan *et al*. [89] investigated the impact of network depths on its accuracy in the domain of large-scale image recognition. The authors have argued that significant improvement can be achieved via increasing the network depth to 16-19 weight layers when using an architecture with very small convolution filters. Based on the findings, the authors achieved the first and second places in 2014 ImageNet Challenge in the localization and classification tracks respectively. Chen and Pock [14] proposed a flexible learning framework based on nonlinear reaction diffusion models for image restoration. Specifically, the reaction diffusion model is dynamic with time-dependent parameters which includes linear filters and influence functions. The advantage is that all the parameters are simultaneously learned from training data. The so called Trainable Nonlinear Reaction Diffusion (TNRD) is capable of dealing with various image restorations tasks by simply changing the reaction force. The authors have reported good performance on Gaussian image denoising, single image super-resolution and JPEG deblocking. In [50], a very deep Convolutional Neural Network is trained to classify ImageNet LSVRC-2010 into 1000 different classes. The training data includes 1.2 million high-resolution images. The authors have reported to achieve 37.5% and 17.0% top-1 and top-5 error rates respectively which claimed to be better than the previous state-of-the-art. The network consists 60 million parameters and 650000 neurons of five convolutional layers along with a final 1000-way softmax. GPU implementation and dropout regularization techniques have been incorporated to boost the speed and reduce over fitting.

In [39], batch normalization technique has been proposed to deal with internal covariate shift by normalizing layer inputs. Internal covariate shift stands for the fact that the distribution of each layer's inputs changes during training deep neural networks since the parameters of the previous layers change. As a result, the training is slowed down and

lower learning rates and good parameter initialization are required. By implementing the proposed batch normalization technique which normalizes each training mini-batch, higher learning rates can be achieved, and possible elimination of the need for dropout. The authors have reported 14 times fewer training steps than the original model. He *et al*. [33] presented a residual network to allow training of deeper neural networks. The layers are reformulated as learning residual functions with references to the layer inputs instead of the unreferenced ones. The authors have argued that this type of network is easier to optimize and achieves better accuracy with increased network depth. The proposed network has a depth of 152 layers which is 8 times deeper than traditional networks. They have reported to achieve 1st place on ILSVRC 2015 classification task with only 3.57% error rate. They also won the 1st place in COCO 2015 competition with 28% relative improvement on the COCO object detection dataset. Szegedy *et al*. [91] proposed the Inception deep Convolutional Neural Network that achieves the new state-of-the-art for classification and detection in 2014 ImageNet Large-Scale Visual Recognition Challenge. The main contribution of the architecture is that they increased network depth and width with improved utilization of the computing resources inside the network based on the Hebbian principle and multi-scale processing. The authors named the incarnation for ILSVRC14 GoogLeNet with a 22 layers design for classification and detection. He *et al*. [32] studied the impact of rectified activation units (rectifiers) on Convolutional Neural Networks in the context of image classification. They firstly proposed a Parametric Rectified Linear Unit (RPeLU) that improves model fitting close to zero extra computational cost. They also derived a robust initialization considering the rectifier nonlinearities. With the proposed technique, it allows to training deep rectified models from scratch and wider network architectures. They have reported 4.94% top-t test error on the ImageNet 2012 classification dataset which is higher than GoogLeNet. It is considered to be the first work to surpass the human-level performance on the ImageNet dataset.

In [21], a family of sub-gradient methods are presented that dynamically incorporate

the geometry of the data from earlier iterations in order to perform gradient-based learning. The methods allow to find tiny features in a very predictive manner. It includes proximal functions to control the steps of gradient with adaptive modification which makes setting the learning rate more simple. The results are guaranteed to be optimal with regularization functions and domain constraints. Kiku *et al*. [46] proposed a residual interpolation to replace the traditional color difference interpolation under the domain of color image demosaicking. The residual is defined as the difference between an observed and estimated pixel value. The proposed residual interpolation is incorporated into the gradient based threshold free algorithm. The authors have shown the proposed demosaicking scheme yields state-of-the-art performance on Kodak and IMAX dataset. Kingma and Ba [49] introduced an algorithm called Adam for first-order gradient-based optimization of stochastic objective function. The algorithm is reported to be computationally efficient and invariant to diagonal re-scaling of the gradients in it's based on adaptive estimates of lower-order moments. It is also suitable for non-stationary objectives with very noisy and sparse gradients. The theoretical convergence properties of the algorithm is analyzed and a bound on the convergence rate is provided.

2.2.3 Related work on Generative Adversarial Network

Goodfellow *et al*. [28] proposed a new framework which contains a generative model G and a discriminative model D. The two models in the framework cooperate in an adversarial manner. The generative model G captures the data distribution, and the discriminative model D computes the probability of whether the generated sample is from the training data. The problem if formulated as a min-max problem for the purpose of generating fake samples which could successfully fool the discriminative model D. Denton *et al*. [17] proposed a generative parametric model that uses a convolutional networks within a Laplacian pyramid framework to generate high quality samples of natural images. A separate GAN is trained at each level of the pyramid. The authors reported 40% of the time their generated

CIFAR 10 samples were classified as real images.

In [52], a GAN is proposed for photo-realistic single image super-resolution. The network is composed of a generative module and a discriminative module. The network is trained on DIV2K dataset. The author argued that even though the generated high resolution image does not outperform others in terms of PSNR, but the perceptual quality of the image is often superior than others. Jolicoeur [44] challenges conventional GANs by arguing that optimization in traditional GANs would decrease the probability of real data being real. The author proposed a relativistic discriminator which estimates the probability that a given sample is real, and reported better performance than traditional GANs.

2.3 Reformulating the Haze Formation Model

We start out discussion from simplified haze observation model:

$$\mathbf{I}(x) = \mathbf{J}(x)t(x) + \mathbf{A}(1 - t(x)), \tag{2.1}$$

where $\mathbf{I}(x)$ stands for observed hazy image, $\mathbf{J}(x)$ is the scene radiance (i.e., unknown clean image to be recovered), $\mathbf{A}$ is the atmospheric light which varies depending on weather conditions, $t(x)$ is the so-called transmission map. In literature, the first term $\mathbf{J}(x)t(x)$ is called *direct attenuation*, and $\mathbf{A}(1 - t(x))$ is called *air light* [66, 93, 31]. The transmission map $t(x)$ is computed by:

$$t(x) = e^{-\beta d(x)}, \tag{2.2}$$

where β is the atmospheric scattering coefficient, a typical range of this value is within $[0.5, 1.8]$. $d(x)$ is scene depth. The larger the scene depth, the heavier the haze. As shown in the above equations, there are two unknowns ($t(x)$ and A) to be estimated in order to recover $\mathbf{J}(x)$. More importantly, the transmission map $t(x)$ is termed along with the target of restoration $\mathbf{J}(x)$ which makes dehazing non-linear and challenging to solve.

We take the analogy from widely-studied problem single image denoising. In denoising, a noisy image can be represented as

$$\mathbf{I}(x) = \mathbf{J}(x) + \mathbf{w}(x) \tag{2.3}$$

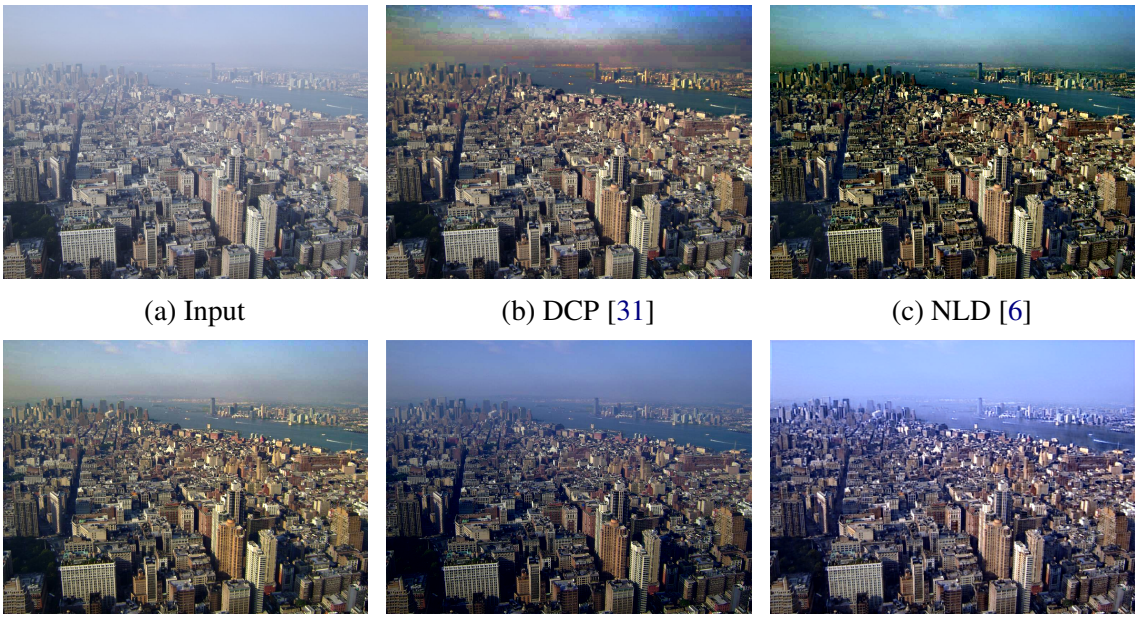
where $\mathbf{I}(x), \mathbf{J}(x)$ denote noisy and clean images respectively, the additive noise term $\mathbf{w}(x) \sim N(0, \sigma_w^2)$ is often assumed to be white Gaussian in the denoising literature. Deep Residual Network has been proposed to solve denoising problem and reported very good performance. By comparing Eq. (2.3) and Eq. (2.1), we propose to reformulate Eq. (2.1) in order to implement DRN:

$$\begin{aligned} \mathbf{I}(x) &= \mathbf{J}(x) + (\mathbf{A} - \mathbf{J}(x))(1 - t(x)) \\ &= \mathbf{J}(x) + \mathbf{r}(x) \end{aligned} \tag{2.4}$$

where $\mathbf{r}(x) = (\mathbf{A} - \mathbf{J}(x))(1 - t(x))$ can be interpreted as a *structured* error term characterizing the nonlinear signal-dependent degradation associated with the hazy effect.

2.4 Recursive Deep Residual Learning for Single Image Dehazing

In previous approaches, estimating transmission map $t(x)$ and atmospheric light A are necessary steps in order to recover scene radiance $\mathbf{J}(x)$. In this work, we propose to bypass estimating $t(x)$ and A, and recover $\mathbf{J}(x)$ by formulating haze as a structural signal-dependent noise $\mathbf{r}(x)$ as shown in Eq. (2.4). Such reformulation allows us to perform an end-to-end optimization without any intermediate steps. Note that our approach does not rely on the haze observation model in Eq. (2.1), in other words, if there exists a more accurate observation model, our approach still works well. Figure 2.2 compares our DRL result against other state-of-the-art approach. Following subsections will introduce the architecture of



(a) Input (b) DCP [31] (c) NLD [6]

(d) MSCNN [79] (e) AOD [54] (f) DRL

Figure 2.2: Visual quality comparison between our proposed DRL and other state-of-the-art methods.

our DRL model and report experimental results.

2.4.1 Network Architecture

As shown in Figure 2.3, our network is composed of three components: the convolution layer (Conv), batch normalization (BN), and rectified-linear unit (ReLU). The first Conv layer consists of 32 feature maps generated by 32 filters sized by $3 \times 3 \times 3$ (since an input image has three color channels) and is followed by the Relu layer which performs $max(0, x)$ operation adding non-linearity to the model. Starting from the second Conv layer (as highlighted by color brown), we use 32 filters of size $3 \times 3 \times 32$ for each Conv layer, followed by BN and ReLU layers. The Conv + BN + ReLU concatenation is repeated for 15 times. Those network parameters such as filter size and network depth were empirically tuned to be nearly optimal. The last Conv layer (as highlighted by color green) includes 3 filters of size $3 \times 3 \times 32$ to calculate the loss between the outputs and training labels.

Based on the analogy between dehazing and denoising, we propose to take a deep

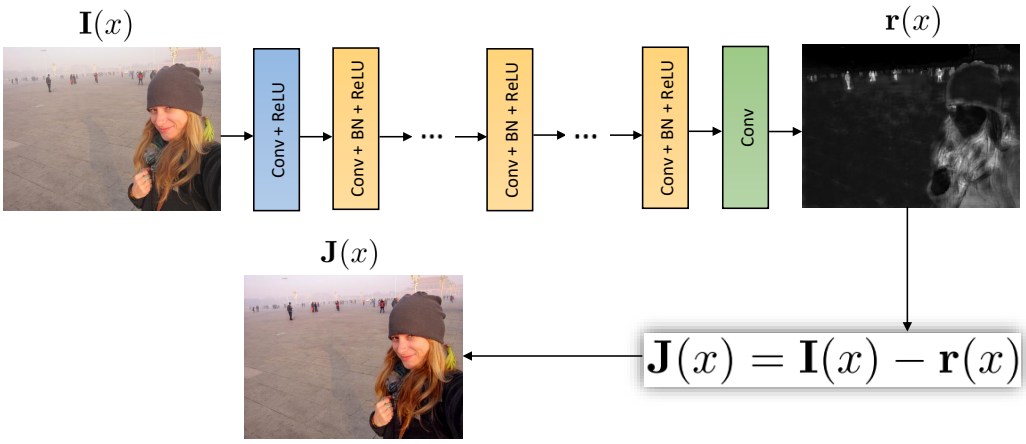


Figure 2.3: The architecture of the proposed DRL network.

residue learning approach toward image dehazing here. More specifically, we aim at learning a nonlinear mapping Ω from $\mathbf{I}(x)$ to $\mathbf{r}(x) = \mathbf{I}(x) - \mathbf{J}(x)$. Then the dehazed image can be recovered via $\mathbf{J}(x) = \mathbf{I}(x) - \mathbf{r}(x)$ where $\mathbf{r}(x) = \Omega(\mathbf{I}(x))$. Similar to [111], we have adopted the following loss function to learn the parameters Θ in the proposed DRL network:

$$\mathcal{L}(\Theta) = \frac{1}{2N} \sum_{k=1}^{N} \|\Omega(\mathbf{I}_k(x); \Theta) - (\mathbf{I}_k(x) - \mathbf{J}_k(x))\|_F^2 \tag{2.5}$$

where $\{(\mathbf{I}_k(x), \mathbf{J}_k(x))\}_{k=1}^{N}$ represents N hazy and original training image (patch) pairs.

2.4.2 Recursive Extention using Fixed-Point Theory

The analogy between dehazing and denoising also motivated us to explore iterative optimization toward the proposed DRL-based dehazing. In denoising, suppose the noisy image is $\mathbf{I}(x)$, the denoised image is represented using a non-linear operator $\Phi^{-1}(\mathbf{I}(x))$, the residual (error term) between the two is $\mathbf{e}(x) = \mathbf{I}(x) - \Phi^{-1}(\mathbf{I}(x))$. When $\mathbf{e}(x)$ is already white Gaussian (i.e., when there is no leftover/residual in $\Phi^{-1}(\mathbf{I}(x))$), we can end the denoising process; Otherwise, a typical strategy is to feed the denoised image back to the input of the algorithm, reducing the leftover in a recursive manner. This process can be applied until $\mathbf{e}(x)$ is white Gaussian, i.e., the fixed point of denoising operator (Ω). This strategy is

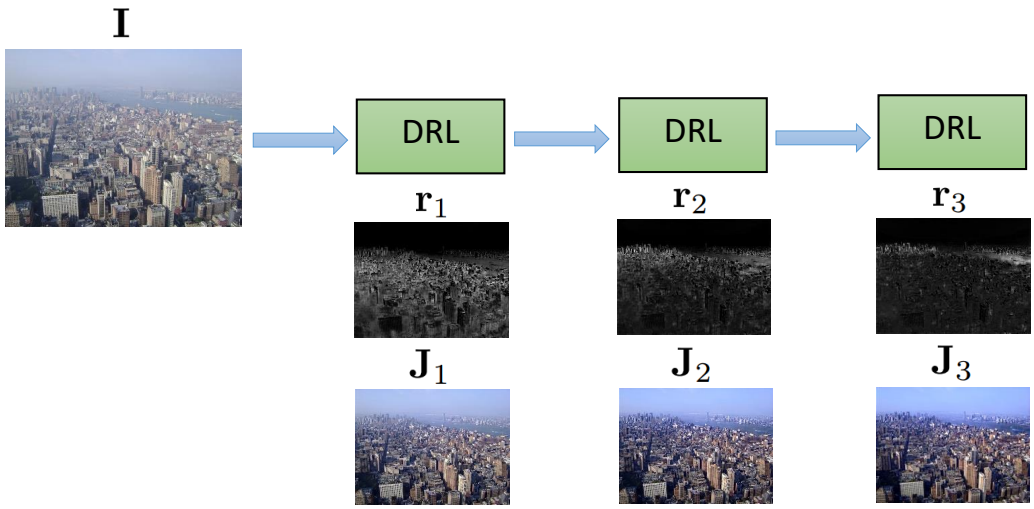


Figure 2.4: Recursive deep residue learning for haze removal: $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ are residuals recovered in the first three iterations; $\mathbf{J}_1$, $\mathbf{J}_2$ and $\mathbf{J}_3$ are recovered scene radiances.

called iterative regularization in literature.

Similarly, we can adopt the same strategy toward the proposed DRL-based dehazing. We feed an input $\mathbf{I}$ into the network, producing a residual map $\mathbf{r}_1$, and obtain an initial estimate of scene radiance $\mathbf{J}_1$. Afterwards, we feed the obtained $\mathbf{J}_1$ to the input of the network again, obtain $\mathbf{J}_2$ and $\mathbf{J}_3$. This process is shown in Figure 2.4. It can be observed that as the number of iteration increases, there is few residual recovered ($\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$), and scene radiance is getting more and more clear ($\mathbf{J}_1$, $\mathbf{J}_2$ and $\mathbf{J}_3$). Depending on the amount of haze in the image, typically the scene radiance recovered after three to five iterations looks the most visually appealing.

2.4.3 Experiments and Results

We have used the NYU-Depth V2 dataset [88] to create synthetic training images. NYU-Depth V2 dataset consists of 1,449 densely labeled indoor color images with ground truth depth information. The raw depth map has been projected and colorized [88] to fill in missing depth labels. Both the color and depth data are of the size 640 $\times$ 480. We pick 1,200 out of 1,449 images to generate training patches, and take the remaining 249 images

as **ImageSet A**; we pick another 21 images from the Middlebury Stereo Datasets [85] as **ImageSet B** and 14 images from NTIRE 2018 dehazing challenge as **ImageSet C**. For each image in the training set, we extract 40 $\times$ 40 patches with stride number being 30; there are 360,064 training patches generated in total.

To simulate synthetic hazy images, the following parameters are used in our experiment. We randomly select $\beta \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5\}$ since any value of β beyond this range could lead to unrealistic haze (too thin or too heavy) and noise amplification [79]. For each of the RGB channel, atmospheric light A is chosen uniformly within the range of $[0.6, 1.0]$. Training labels are generated using Eq. (2.4). During the training process, the weights of each convolution layers are randomly initialized by Gaussian variables. The number of epocs is set to 100; the learning rates for the first 60 and the remaining 40 epocs are set to 0.001 and 0.0001 respectively. We have selected Stochastic Gradient Descent (SGD) as the solver with a momentum parameter of 0.9. The network is trained on a PC with an Intel i7-4790k processor and a Nvidia GeForce Titan GPU leading to the total training time of about 15 hours.

Experimental Results on Synthetic Data and Real-world Images

In order to show the effectiveness of the proposed DRL, we compare our network with several state-of-the-art dehazing methods: Dark Channel Prior (**DCP**) [31], Boundary Constrained Context Regularization (**BCCR**) [65], Visual Artifact Suppression via Gradient Residual Minimization (**VASGRM**) [13], Non-Local Image Dehazing (**NLD**) [6], **MSCNN** [79], **DehazeNet** [11] and **AOD** [54]. The first four methods, including DCP, BCCR, VASGRM and NLD, are traditional model-based approaches, and the remaining ones are learning-based approaches. Two objective image quality metrics are used in our comparison: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) [98]. Table 2.2 and 2.3 shows the average PSNR and SSIM comparison results on ImageSet A and B respectively. Overall, we have observed that learning-based approaches, such as

Table 2.2: Average PSNR and SSIM results on ImageSet A.

Metrics	DCP	BCCR	VASGRM	NLD	MSCNN	DehazeNet	AOD	**DRL**
PSNR	17.94	14.77	16.25	15.97	19.23	15.35	18.36	**21.7**
SSIM	0.86	0.81	0.83	0.77	0.86	0.76	0.85	**0.92**

Table 2.3: Average PSNR and SSIM results on ImageSet B.

Metrics	DCP	BCCR	VASGRM	NLD	MSCNN	DehazeNet	AOD	**DRL**
PSNR	15.82	14.49	15.57	16.33	18.16	19.43	17.51	**21.41**
SSIM	0.81	0.77	0.81	0.79	0.84	0.85	0.85	**0.86**

MSCNN, DehazeNet, and AOD, produce slightly better PSNR/SSIM results than model-based approaches; while our DRL outperforms all others by a large margin. We believe that dramatic performance improvement is jointly contributed by deep residue learning and fixed-point iterations.

Figure 2.5 and 2.6 shows the performance of our method compared against AOD [54] on ImageSet A and B. Since the NYU-Depth V2 dataset has been collected under indoor environment, most of the images have busy background with furniture and objects. Our method can handle such busy background very well - the scene radiance recovered maintains good sharpness with haze residual being close to zero. Our dehazed results are convincingly better than those produced by AOD on this data set. The main challenge of dehazing on ImageSet B is to recover rich details around sharp edges and vivid colors in those Middlebury images. Figure 2.5 and 2.6 shows our method can more faithfully maintain important image structure such as corners and edges and better recover the color fidelity than previous work of AOD. Figure 2.7 compares our approach against others on NTRIE 2018 dehazing challenge dataset. Our approach often recovers scene radiance with faithful colors than others.

We have also compared our method with seven competing dehazing approaches on real-world hazy images as shown in Figure 2.22 (a). This set of images - containing a large variation of scene content such as portrait, landscape and architecture - come from our

Table 2.4: Comparison of average running time in seconds.

Method	Time	Platform
DCP [31]	8.3	Matlab
BCCR [65]	1.6	Matlab
VASGRM [13]	18.1	Matlab
NLD [6]	2.4	Matlab
MSCNN [79]	0.98	Matlab
DehazeNet [11]	1.4	Pycaffe
AOD [54]	0.53	Pycaffe
DRL	**0.87**	Matlab

own collection of real-world images that have been used in previous studies. These images contain both heavy and thin haze, shallow and large depth field, coarse and fine details, which reflect the diverse challenges in the real world. As we can see from Figure 2.22, our proposed technique has achieved at least comparable (and often superior) visual quality to other competing approaches.

Running Time Comparisons

Table 2.4 shows the comparison of running time (in seconds) among eight competing dehazing techniques. The results are obtained by running each method on ImageSet A which includes 249 images and taking their average. We have found that our method is relatively fast with 0.89 second per image (it is only marginally slower than AOD [54]). We believe this difference is mainly because of the platform, since Pycaffe is better optimized than Matlab in terms of implementing deep Convolutional Neural Networks.

2.4.4 Conclusion

We propose a novel reformulation of haze observation model toward single image dehazing. Our reformulation allows us to model haze as a signal-dependent residual, which could be recovered via Deep Residual Network. We also propose to remove haze recursively using via iterative optimization using fixed-point theory. Compared with conventional wisdom,

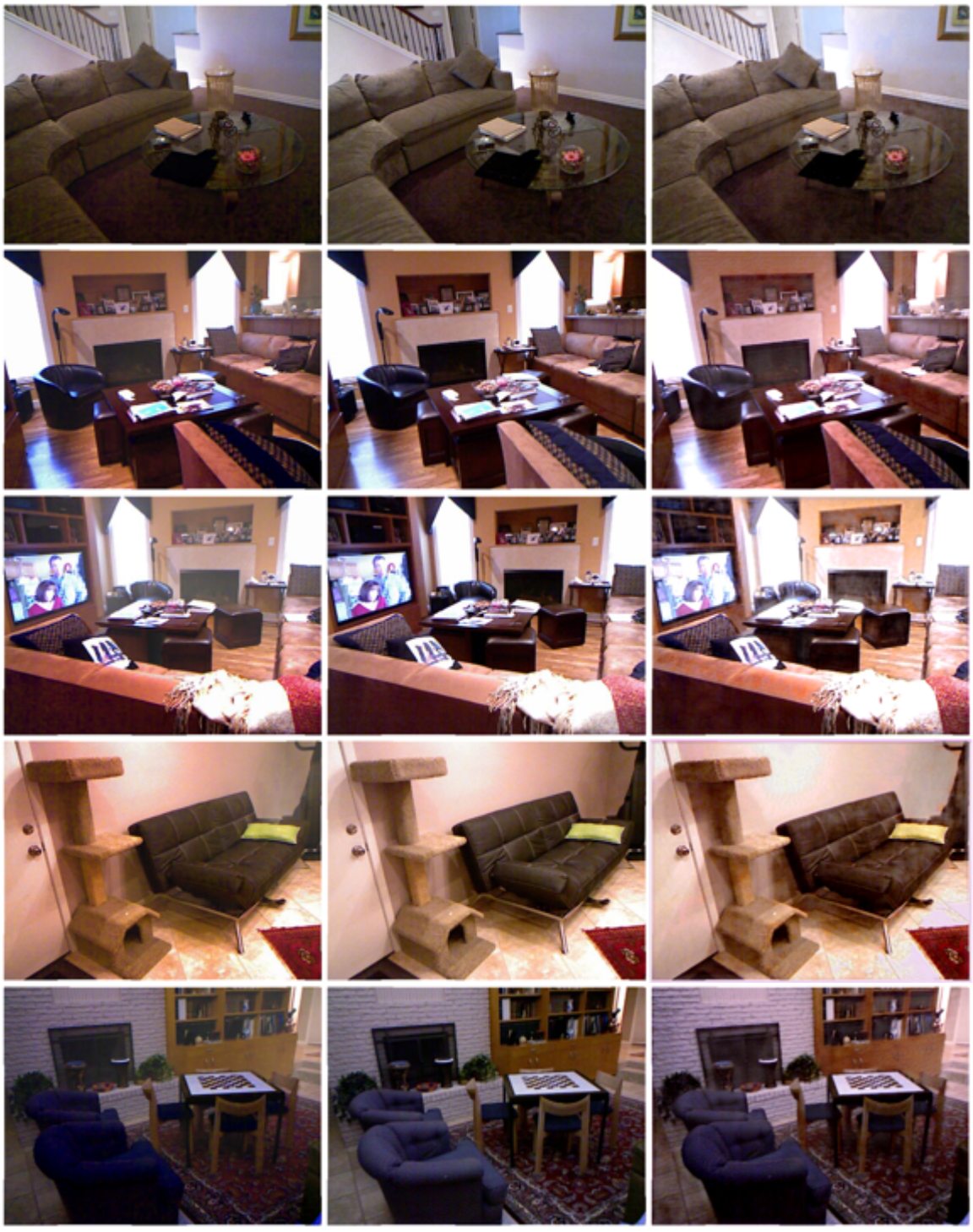


Figure 2.5: Dehazing results on ImageSet A. Left: AOD results [54]. Middle: original images. Right: outputs of DRL.

Figure 2.6: Dehazing results on ImageSet B. Left: AOD results [54]. Middle: original images. Right: outputs of DRL.

(a) Input (b) NLD [6] (c) VASGRM [13]

(d) BCCR [65] (e) AOD [54] (f) DRL

(g) Input (h) NLD [6] (i) VASGRM [13]

(j) BCCR [65] (k) AOD [54] (l) DRL

Figure 2.7: Visual quality comparison on NITRE 2018 challenge images.

Figure 2.8: Dehazing results on real-world images.

our approach does not relay on the haze observation model, and it bypasses the estimation of transmission map and atmospheric light. The scene radiance is recovered in and end-to-end fashion. Extensive experimental results show that our approach restores high objective quality images, and the visual quality is often superior than others.

2.5 Perceptually Optimized Generative Adversarial Network for Single Image Dehazing

We highlight a few limitations of the DRL-based dehazing approach: The result is debatable from a perceptual point of view; We found it is often burdensome to decide how many iterations is needed toward the recursion; There is some halo effect around large scene discontinuities. To overcome these limitations, we proposed to incorporate another two modules into dehazing. First, we add a discriminative module with adaptive perceptual loss to further improve visual quality of dehazed image. Second, a novel halo removing module based guided filtering is implemented to suppress the halo effect. As a result, we observed substantially improvement in visual quality of dehazed image, as the color is more faithful and there is no noticeable halo effect.

2.5.1 Limitations of DRL-based Dehazing

We have advocated a DRL-based dehazing network based on reformulating the haze observation model, and the optimization is done in an end-to-end fashion. As shown in Figure 2.9, we noticed a few shortcomings of DRL-based approach when conducting the experiment:

- The perceptual quality needs further improvement: previous DRL-based approach is optimized using $\mathcal{L}2$ loss which simply computes pixel-wise differences between dehazed and ground-truth patches. Even though the objective result looks good with respect to PSNR, the perceptual quality needs to be improved. We will further describe in detail the subtle differences between objective and perceptual quality of image later.

- As discussed in Subsection 2.4.2, we propose to remove haze recursively by feeding the output of dehazing back to the input of the network. Such technique could poten-

tially produce better looking scene radiance. But the side effect is how to determine the appropriate number of iterations. Even though we found typically it takes three to five iterations, but it is still impossible to run the algorithm fully automatically since every input hazy image is different. To overcome this limitation, we propose to replace the manual external recursion with an internal recursion (i.e., the so-called residual in residual), we will further discuss the network architecture in detail.

- Notorious halo effect: As discussed in Section 2.1, haze is an inconsistent behavior since it depends on scene depth. This brings challenges toward learning-based approach: one needs to create large amount of training data covering both light and heavy haze conditions. As a result, we noticed halo effect around large scene depth discontinuities. These effect vastly impact the visual quality of dehazed image.

2.5.2 The Proposed Approach

We propose a GAN-based approach which aims at optimizing perceptual quality of images. Our GAN is composed of two components: namely a generative module G to remove haze from input and a discriminative module D to judge the output of generative module. We emphasis that out G module adopts the residual in residual architecture that replaces the external recursion proposed in Subsection 2.4.2 to alleviate the burden of choosing the number of recursions. We also propose a novel halo removing module based on guided filter. It smoothies the residual map produced by G and D using the original input as guidance. Following parts will introduce each component of our GAN-based dehazing approach.

Why GANs: The Magic of Distortion v.s. Perception

Conventional wisdom believed that the lower the distortion (e.g., PSNR, SSIM, IFC, VIF), the better the visual quality. Thus, most of algorithms aim at maximizing some distor-

Figure 2.9: Limitation of DRL-based dehazing. The left column shows the input image, the right column shows the dehazing results with DRL. Notice that the results are often not perceptually pleasing, and there is some halo effect around large scene discontinuities.

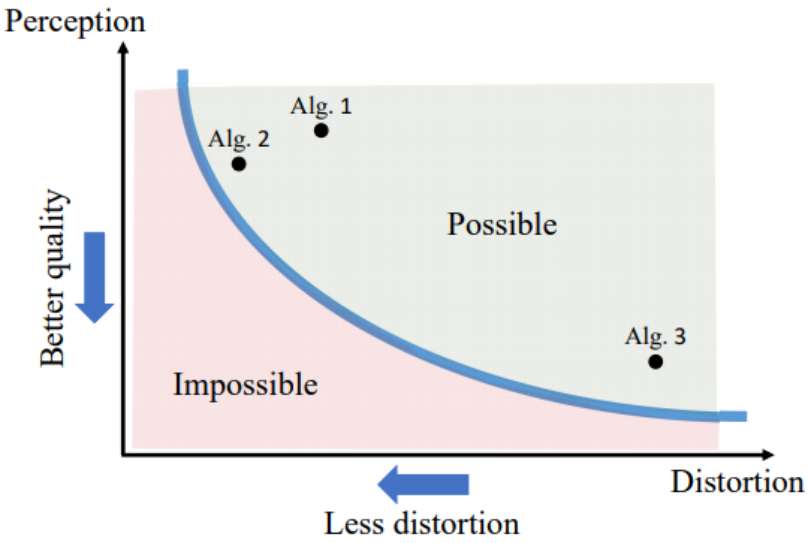


Figure 2.10: Figure in [10] shows the perception-distortion trade-off. The authors argued that there is an unattainable region when one wants to optimize both distortion and perceptual quality. They also claimed that it's applicable to only optimize one of the two.

tion measure. As shown in Figure 2.10, Blau and Michaeli [10] proved the odd of the conventional wisdom by showing that there is an unattainable region when one wants to optimize both distortion and perceptual quality. They claimed that it's applicable to only optimize one of the two. In image restoration, the goal is to estimate an image x from its degraded version y. There is substantial difference within the motivation when performing optimization toward distortion or optimization: distortion stands for the dissimilarity between the reconstructed image $\hat{x}$ and the original image x, and perceptual quality refers to the visual appearance of $\hat{x}$, regardless of how similar $\hat{x}$ looks like the to x [10]. They also argued that generative adversarial networks provide a principled way to approach the perception-distortion bound.

In single image dehazing, we argue that it is often more favorable to prioritize perceptual quality, rather than distortion measure. Thus, as an extension of our DRL-based dehazing, we propose to add a discriminative module with adaptive perceptual loss function. We show in the experiment that the perceptual optimized dehazing results has much better visual quality. An shown in Figure 2.11, our approach is composed of three components, a generative module with residual learning structure based on our novel reformulation, a

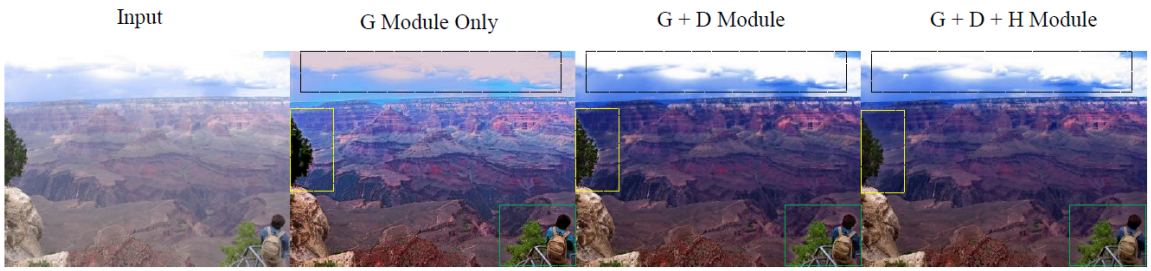


Figure 2.11: Perceptual optimization achieved by the proposed GAN including a generative network G, a discriminative network D, and a post-processing module H.

discriminative module with VGG and adversarial loss, and a post processing module in order to remove the halo effect.

Generative Module with Residual-in-Residual Architecture: External Recursion v.s. Internal Recursion

As discussed in Subsection 2.4.2, we propose to remove haze recursively by feeding the output of dehazing back to the input of the network. The advantage of this technique is the recursively dehazed image often looks better that non-recursive version. However, deciding the proper number of recursion iterations brings complexity to the algorithm. Thus, in the generative module of GAN-based dehazing, we propose an internal recursion structure (the so-called residual in residual) to replace the previous external one. This alleviate the trouble of choosing recursion iterations, yet still gives faithful results since the features are more rich.

Figure 2.12 shows the architecture of our recursive deep residual learning module with corresponding filter size (f) and the number of feature channels (c). The module takes a hazy patch ($50 \times 50 \times 3$) as input, followed by Convolution (Conv) and Rectified Linear Unit (ReLU) layer with 64 feature channels and 3×3 filter size. The residual block includes 16 sub-blocks. Each sub-block is composed of Conv, Batch Normalization (BN), Relu, Conv, and a Element-wise (Elti) Subtraction layer. The Elti layer takes the input from last sub-block, subtracts the residual recovered in the current sub-block, and return a less-hazy

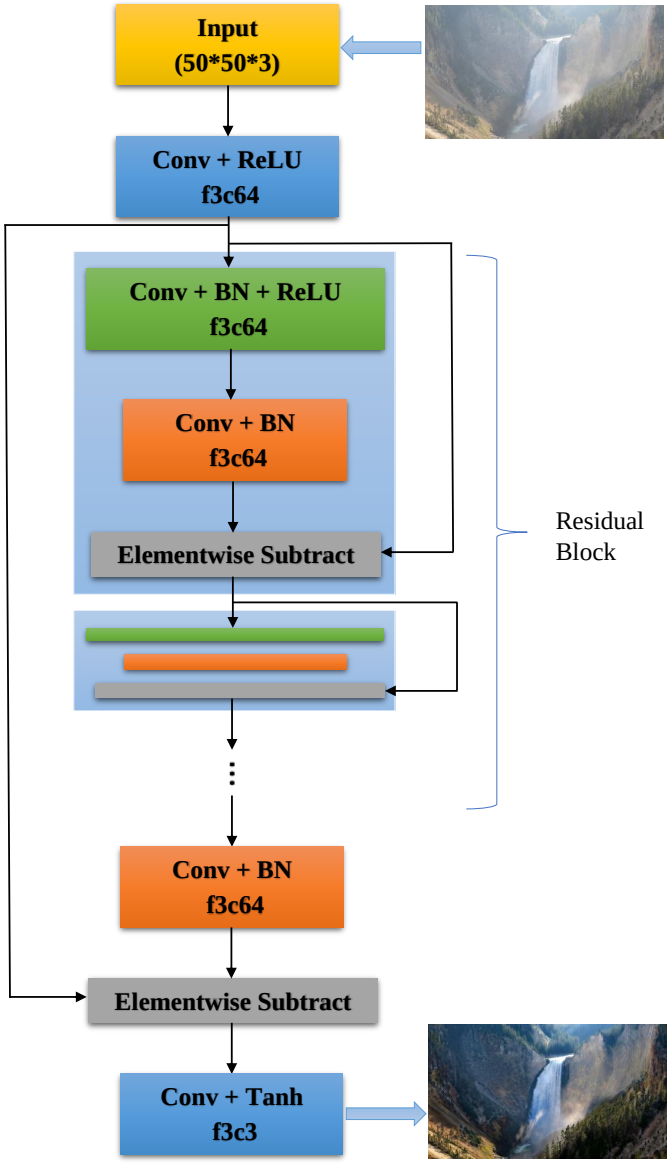


Figure 2.12: Architecture of a deep residual network with corresponding filter size (f) and the number of feature channels (c).

patch in a progressive manner. The last Elti layer performs a pixel-wise subtraction of the input and output of residual block followed by another Conv layer with a hyperbolic tangent activation function.

Discriminative Module and Loss Funtions

As mentioned above, it is usually difficult to address the issue of visual quality assurance in previous transmission-map-first approaches. Bypassing the estimation of transmission map makes it possible to leverage the idea of generative adversarial networks (GAN) from image synthesis [17] and super-resolution [52] to image dehazing. The basic idea behind GAN is to introduce a discriminative network as a judge telling whether the output of generative network is real or fake. Under the context of dehazing, G produces dehazed image patches and D classifies them as dehazed (fake) and haze-free (real). The goal of adversarial learning is for G to produce dehazed patches that can fool D (dehazed patches are visually indistinguishable from haze-free ones). We have followed the discriminative architecture guidelines proposed in [52, 78] which contains Conv layers with 64, 128, 256, and 512 channels. The last Conv layer is followed by two dense layers and a sigmoid activation function to classify the dehazed and haze-free patches.

We have adopted the perceptual loss function proposed in [52] which is a weighted sum of MSE, VGG loss, and adversarial loss respectively:

$$l = w_1 l_{MSE} + w_2 l_{VGG} + w_3 l_{Adv}, \tag{2.6}$$

where w_i (*i=1,2,3*) controls the weight of each term. The pixel-wise MSE loss is given by:

$$l_{MSE} = \frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} (\mathbf{J}^*_{x,y} - \mathbf{J}^G_{x,y})^2, \tag{2.7}$$

where $\mathbf{J}^*$ denotes the ground truth (haze-free image) and $\mathbf{J}^G$ denotes the dehazed one by the G module. The VGG loss computes the Euclidean distance between the feature maps of $\mathbf{J}^*$ and $\mathbf{J}^G$:

$$l_{VGG} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(\mathbf{J}^*)_{x,y} - \phi_{i,j}(\mathbf{J}^G)_{x,y})^2, \tag{2.8}$$

where $W_{i,j}$ and $H_{i,j}$ denotes the dimensions of extracted feature maps. Finally, the adversarial loss can be written as:

$$l_{Adv} = \sum_{n=1}^{N} -\log\left(\mathbf{J}^{D}\right), \tag{2.9}$$

where $\mathbf{J}^{D}$ is the probability that the reconstructed image is haze-free.

To the best of our knowledge, previous GANs mostly use loss functions with fixed weights for each module such as [28, 17, 78, 37, 60]. Toward the objective of perceptual optimization, we propose an adaptive perceptual loss function tailored to fit the severity of haze in an image. The rationale is that the process of dehazing has to deal with various uncertainty factors such as direct attenuation and airlight in the image degradation model. Since the attenuation term $\mathbf{J}(x)t(x)$ dominates the thickness of haze during the degradation, it is natural to adaptively choose the weights of loss function based on the attenuation term. That is, we can adjust w_1, w_2, and w_3 based on the amount of attenuation controlled by β (large β corresponding to heavy haze). More specifically, we propose to use larger w_1 under heavy haze situation (i.e., more emphasis on haze removal) and larger w_3 under light haze condition (i.e., more emphasis on quality assurance).

Removing the Halos

It has been widely recognized that dehazed images have the tendency of producing various halo-like artifacts (e.g., ringing reduction [26], anti-halation enhancement [54], block halo suppression [109]). We have also empirically observed that the proposed GAN-based dehazing sometimes suffer from noticeable halo-like artifacts especially around the areas of large depth discontinuities (i.e., rapid change of attenuation) as shown in Figure 2.13. We have also found that using larger filters (e.g., 5×5, 7×7, or a combination of filters with different sizes) tend to make the artifacts more serious (the so-called block halo problem [109]).

(a) Input $\mathbf{I}$ (b) Initial $\mathbf{J}_1$ (c) Residual $\mathbf{r}_1$ (d) Refined residual $\mathbf{r}_2$ (e) Final $\mathbf{J}_2$

Figure 2.13: The proposed post-processing module for halo suppression. The initial dehazed image $\mathbf{J}_1$ is the input. We first obtain $\mathbf{r}_1$ via elemental-wise subtraction $Elti(\mathbf{I},\ \mathbf{J}_1)$. Then refined residual $\mathbf{r}_2$ is obtained by applying guided filtering to $\mathbf{r}_1$ ($\mathbf{I}$ is the guidance). Finally, $\mathbf{J}_2$ is recovered from $Elti(\mathbf{I},\ \mathbf{r}_2)$.

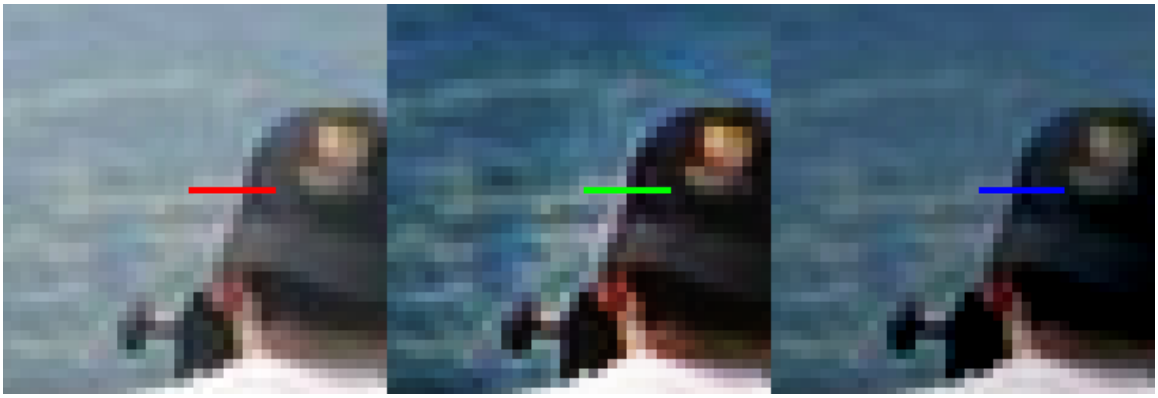
To suppress the potential halo-like artifacts, we propose to include a guided-filtering based post-processing module. Guided filtering was first proposed in [30] and its effectiveness on refining the estimated transmission map has been well documented in the literature (e.g., [74]). Here we suggest a novel application of this powerful tool into refining the residual map as a post-processing strategy. Guided filtering assumes the following linear relationship between the guidance I and output q:

$$q_i = \alpha_k I_i + b_k, \forall i \in \omega_k, \tag{2.10}$$

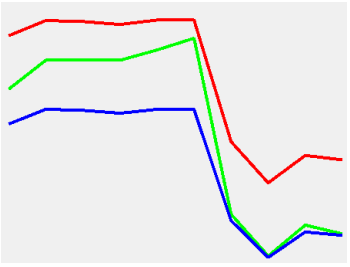
where (a_k, b_k) are some constant linear coefficients. To compute these coefficient, one needs to minimize a cost function characterizing the difference between q and the input p in a window ω_k [30]:

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon \alpha_k^2) \tag{2.11}$$

As shown in Figure 2.13, our novel application of guided filtering into halo removal consists of three steps. First, We obtain raw residual $\mathbf{r}_1$ via elemental-wise subtraction $Elti(\mathbf{I},\ \mathbf{J}_1)$ where $\mathbf{J}_1$ is recovered by applying G and D module on the hazy image $\mathbf{I}$. Second, the *refined* residual $\mathbf{r}_2$ is obtained by applying a guided filter using $\mathbf{I}$ as the guidance and $\mathbf{r}_1$ as the input image. Finally, refined image estimation $\mathbf{J}_2$ is recovered from $Elti(\mathbf{I},\ \mathbf{r}_2)$. To better illustrate how the proposed post-processing module works, we have



(a) From left to right: $\mathbf{I}$, $\mathbf{J}_1$, and $\mathbf{J}_2$



(b) 1-D illustration for halo removing.

Figure 2.14: Visual comparison of zoomed portions in Fig. 2.13 and the corresponding 1D intensity profiles.

used a toy example as shown in Figure 2.14. Figure 2.14a shows the zoomed comparison of $\mathbf{I}$, $\mathbf{J}_1$, and $\mathbf{J}_2$ around the head region (where large scene depth discontinuity occurs) respectively; and figure 2.14b shows the corresponding 1-D plot of intensity pixels. It can be clearly observed how the overshooting estimate by our generative network gets corrected by guided filtering.

2.5.3 Experimental Results

Datasets and Implementation Details

Preparation of training data plays an important role in deep learning-based approaches. Previous works such as MSCNN [79] and AOD [54] have used the NYU-Depth V2 [88] dataset where color and depth images are captured by Microsoft Kinect. In view of the limited image quality of the NYU-Depth V2 dataset, we have taken 799 image from the DIV2K [2]

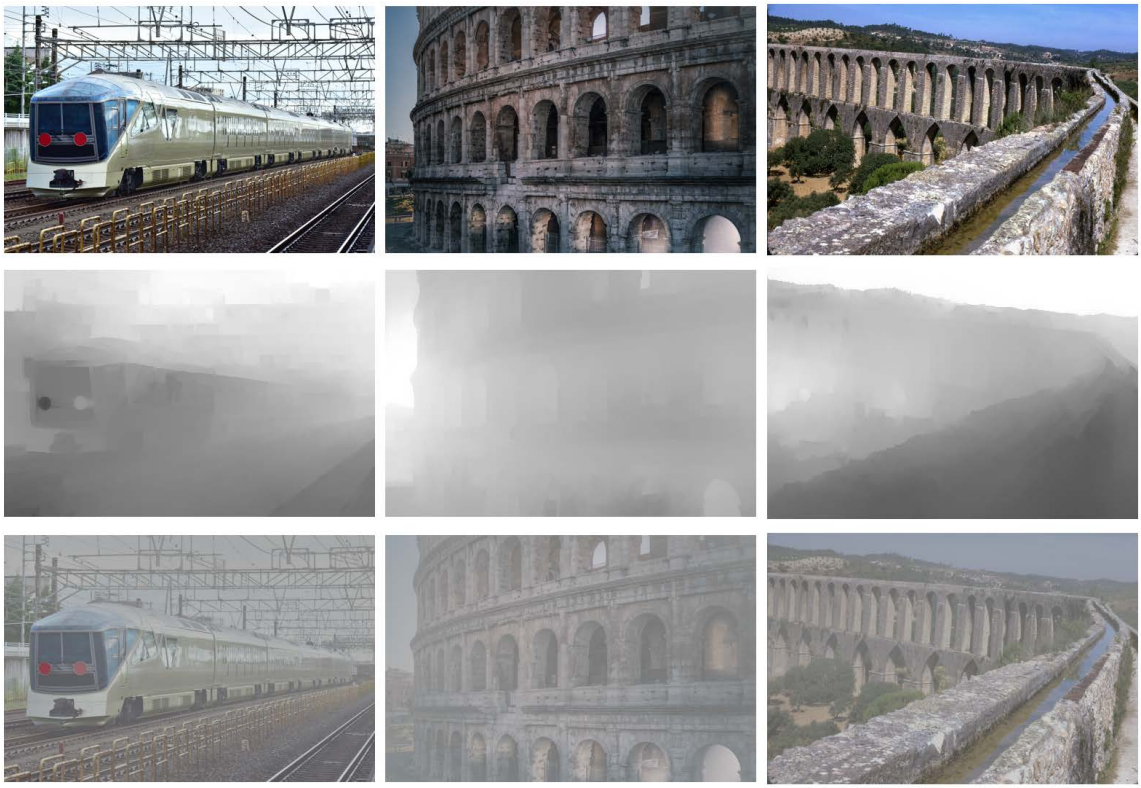


Figure 2.15: Creation of training dataset. From top to bottom row: original image from DIV2k [2] dataset, depth map computed using [59], and the hazy images generated by Eq. (2.1).

dataset with high quality images (originally constructed for image super-resolution). To obtain the corresponding depth images, we have borrowed a deep CNN-based approach of learning depth images from single monocular images [59]. Figure 2.15 shows some examples of the learned depth maps for the preparation of training dataset. To simulate synthetic hazy images, the following parameters are used in our experiments: we have randomly selected attenuation parameter $\beta \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5\}$ since any value beyond this range could lead to unrealistic haze (too thin or too heavy) and unwanted noise amplification [79]. For each of the RGB channel, atmospheric light A is chosen uniformly within the range of $[0.7, 1.0]$.

The test dataset consists of both synthetic and real-world hazy images. Similar to the previous work [54], the synthetic test data contains 100 images from the DIV2K dataset as **ImageSet A** and 21 images from the Middlebury Stereo Datasets [85] as **ImageSet B**;

additionally, we pick another 31 real-world images as **ImageSet C**. During the training process, the weights of each convolution layers are randomly initialized by Gaussian variables. The patch size is 50×50; the number of epochs is set to 100; the learning rates for the first 50 and the remaining 50 epochs are set to 0.001 and 0.0001 respectively. We have selected Adam optimizer with Beta1 parameter being 0.9. The network is implemented using TensorFlow and trained on a PC with an Intel i7-4790k processor and a Nvidia GeForce Titan GPU.

Effectiveness of Each Module

We have designed a series of comparative studies to facilitate the illustration of each module in the proposed approach. First, we want to demonstrate the effectiveness of introducing a *discriminative* network on visual quality assurance. Figure 2.16 shows the dehazing results on a pair of real-world images without and with a discriminative network. It can be observed that dehazing with a generative network only tends to remove haze over aggressively, especially in the background where there is heavy haze. The undesirable consequence is that some part of the foreground (e.g., trees and mountains) becomes unnaturally dark. By contrast, the inclusion of a discriminative network makes the dehazed images visually more pleasant as shown in the right column of Figure 2.16. This is due to the perceptual loss function helps to ensure that the dehazed images are as close as possible to real haze-free images.

Second, we need to justify the effectiveness of the proposed *post-processing* module. Even though with a discriminative network, we have observed that noticeable halo artifacts could occur around areas with large scene depth discontinuities (e.g., within 4-8 pixels away from the boundary between foreground and background). Meantime, the larger the filter size, the more serious the halo artifacts become which agrees with the observation made in [109]. Figure 2.17 shows that the comparison of dehazed images before and after the proposed post-processing module. It can be clearly seen that undesirable halo artifacts

Figure 2.16: Discriminative network improves the visual quality of dehazed images. Left: input; middle: dehazing with G module optimized using MSE only; right: dehazing with both G and D module optimized for human perception.

in highlighted dashed areas have been successfully suppressed after post-processing.

Third, we aim at illustrating the benefit of *adaptive* perceptual loss function in GAN-based dehazing. In our experiments, we increase w_1 from 0.95 to 1 and w_2 from 0.000001 to 0.000002 respectively; and decrease w_3 from 0.002 to 0.001 as the attenuation parameter β varies within its operational range (as shown in Figure 2.18). Such adaptive weight setting is compared against a fixed setting ($w_1 = 1, w_2 = 0.000001, w_3 = 0.002$). Figure 2.19 shows the comparison of dehazed images between fixed and adaptive weights under varying haze conditions. It can be verified that the proposed strategy of adaptive weights are capable of more effectively removing heavy haze while preserving the visual quality under light haze conditions (i.e., to achieve the objective of perceptual optimization).

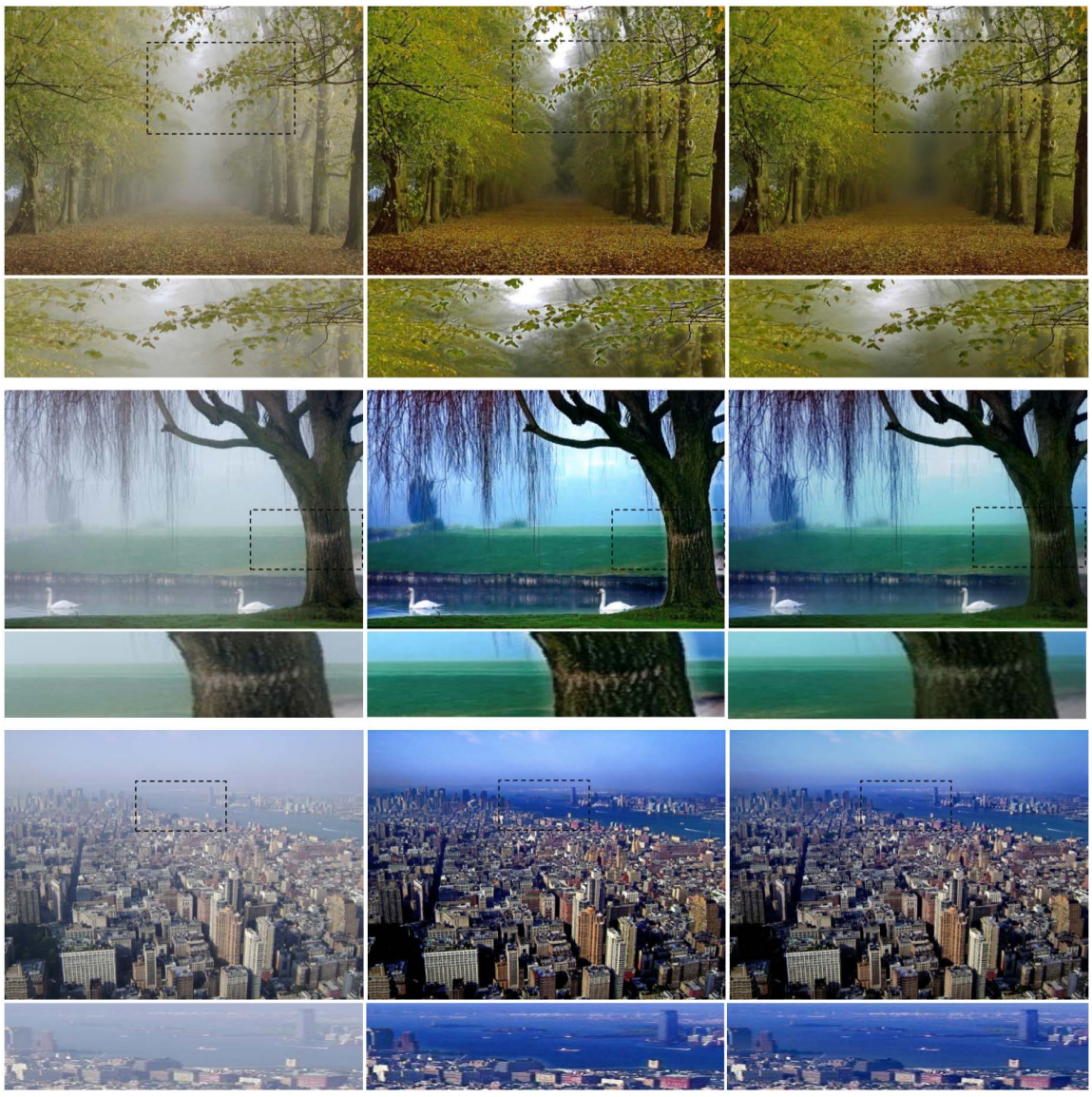


Figure 2.17: Post-processing module improves the visual quality of dehazed images. From left to right: input, dehazing results without and with the post-processing module.

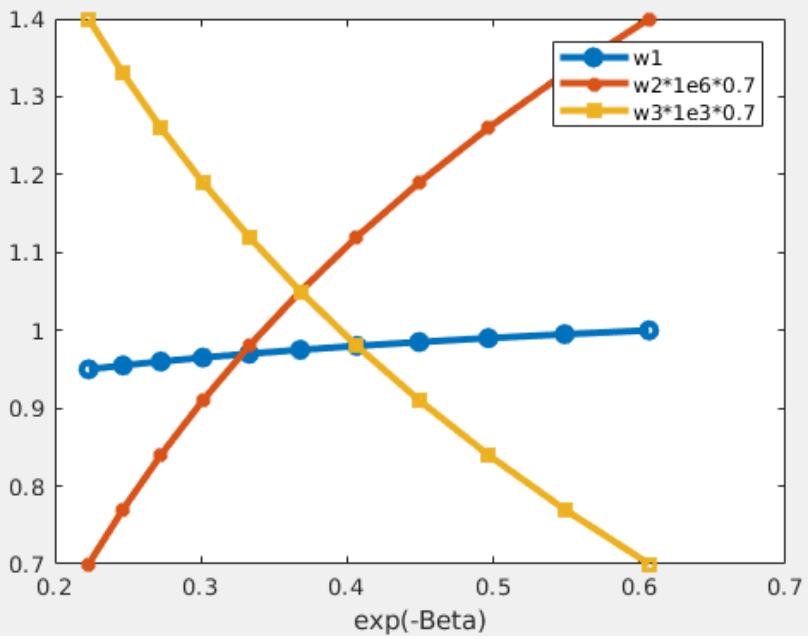


Figure 2.18: The adjustment of weights in Eq. (2.6) based on the severity of attenuation (w_1, w_2, w_3 correspond to blue, red and yellow respectively).

Comparison against State-of-the-Art on Synthetic Images

We have compared our POGAN-based image dehazing with several state-of-the-art dehazing methods including: Dark Channel Prior (**DCP**) [31], Non-Local Image Dehazing (**NLD**) [6], Boundary Constrained Context Regularization (**BCCR**) [65], Multi-Scale CNN (**MSCNN**) [79], **DehazeNet** [11], and **AOD** [54]. The first three methods, including DCP, BCCR and NLD, are state-of-the-art model-based methods, and the last three are leading learning-based methods. Two objective image quality metrics have been used in our comparison: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM). We have conducted experiments on both ImageSet A and B with $\beta \in \{0.75, 1, 1.25, 1.5\}$ and $A \in \{0.7, 0.8, 0.9, 1\}$. Table 2.5 shows the average PSNR and SSIM comparison results on ImageSet A and B respectively. Overall, it can be observed that DCP and AOD respectively outperform others on ImageSet A and B respectively; however, only ours can perform equally well on both datasets. In fact, our PSNR/SSIM performance only slightly falls behind the best method on both data sets.

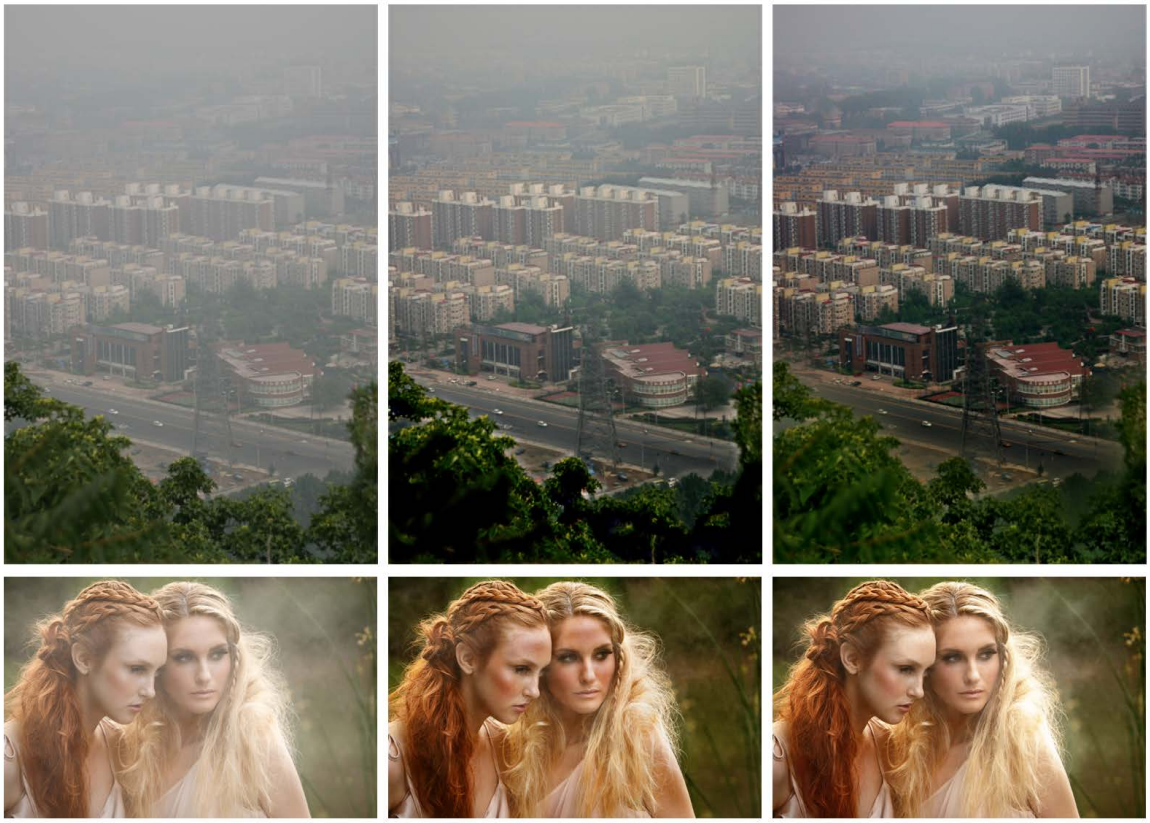


Figure 2.19: Benefit of adaptive perceptual loss function in GAN-based dehazing. From left to right: input, dehazing results of GAN with fixed weights, and dehazing results of GAN with adaptive weights (top: heavy haze; bottom: light haze).

More importantly, we note that objective measures such as PSNR and SSIM often do not faithfully reflect the subjective quality of an image [68]. A convincing counterexample was reported in [52] where the PSNR value of a perceptually much better image is dramatically (over $2dB$) lower than the benchmark; we note that this is also the reason for introducing adversarial loss in Eq. (2.6). As shown in Figure 2.20 and 2.21, it can be observed that our approach generates dehazing results with less color distortion and visually look more favorably than DCP/AOD (closer to the ground-truth - the middle column). We attribute the visual superiority with slightly lower PSNR/SSIM to the design of the perceptual loss function. As shown in Eq. (2.6), the perceptual loss function not only includes the MSE loss, but also the VGG and adversarial loss which optimize the visual quality of an image to human perception.

Comparison against State-of-the-Art on Real-World Images

We have also compared our method against six state-of-the-art dehazing approaches on real-world hazy images (ImageSet C) as shown in Figure 2.22. These approaches include the most recent work on dehazing in 2018 such as **DCPDN** [110],**GFN** [80] and **cGAN** [57]. This set of images contains large variations of scene depths and haze thickness as well as diverse scene structures such as portrait, landscape and architecture. Most of the images in ImageSet C have been included for evaluation purpose in previous studies of single image dehazing. We summarize the superiority of the proposed POGAN-based dehazing as follows: 1) it is effective at removing heavy haze in the presence of large scene depth variations such as the second row in Figure 2.22; 2) it significantly outperform other competing methods in terms of restoring color fidelity and vividness such as the second to the last row in Figure 2.22.

Table 2.5: Comparison of PSNR and SSIM values. Bold denotes the best in it's corresponding row.

Parameters		DCP [5]		NLD [20]		BCCR [19]		MSCNN [7]		DehazeNet [6]		AOD [25]		Ours	
DIV2K															
$\beta = 0.75,\ A =$	0.7	21.73	0.895	20.24	0.871	18.01	0.832	20.21	0.874	17.51	0.751	17.07	0.892	21.55	0.893
	0.8	22.26	0.892	19.99	0.865	18.64	0.829	21.36	0.872	15.99	0.711	19.84	0.923	21.79	0.912
	0.9	22.35	0.883	20.19	0.869	18.94	0.821	21.06	0.863	14.42	0.655	23.11	0.936	21.63	0.893
	1	22.04	0.874	19.18	0.858	18.95	0.812	19.57	0.851	13.05	0.594	22.91	0.932	20.96	0.879
Average		**22.09**	0.886	19.91	0.865	18.63	0.823	20.55	0.865	15.24	0.677	20.73	**0.921**	21.48	0.894
$\beta = 1,\ A =$	0.7	21.47	0.884	19.96	0.866	17.89	0.817	20.46	0.856	16.51	0.701	17.11	0.865	20.61	0.909
	0.8	22.03	0.878	20.62	0.871	18.58	0.815	20.66	0.851	14.88	0.653	19.88	0.891	20.13	0.911
	0.9	22.13	0.871	19.85	0.868	18.77	0.806	19.23	0.836	13.16	0.582	20.98	0.893	21.22	0.891
	1	21.28	0.861	18.76	0.846	18.52	0.791	17.03	0.813	11.62	0.513	18.31	0.877	20.96	0.886
Average		**21.72**	0.873	19.79	0.862	18.44	0.807	19.34	0.839	14.04	0.612	19.07	0.881	20.73	**0.899**
$\beta = 1.25,\ A =$	0.7	21.35	0.873	19.67	0.865	17.74	0.801	19.91	0.825	14.65	0.659	16.82	0.812	20.74	0.812
	0.8	21.93	0.872	20.16	0.864	18.43	0.802	19.02	0.816	14.05	0.61	18.88	0.832	20.51	0.834
	0.9	21.84	0.862	19.82	0.861	18.45	0.792	17.21	0.798	12.19	0.524	18.22	0.827	19.36	0.856
	1	20.25	0.852	17.91	0.829	18.05	0.776	14.79	0.769	10.49	0.447	14.92	0.804	20.86	0.883
Average		**21.34**	**0.864**	19.39	0.854	18.16	0.792	17.73	0.802	12.84	0.561	17.21	0.818	20.36	0.846
$\beta = 1.5,\ A =$	0.7	21.13	0.863	19.34	0.853	17.67	0.786	18.64	0.788	14.92	0.621	16.35	0.748	19.44	0.822
	0.8	21.78	0.866	20.09	0.869	18.24	0.784	17.44	0.771	13.45	0.571	17.55	0.761	18.36	0.815
	0.9	21.18	0.855	19.02	0.842	18.11	0.779	15.41	0.751	11.44	0.482	15.93	0.751	18.27	0.804
	1	18.89	0.842	17.52	0.838	17.55	0.766	13.04	0.718	9.91	0.409	12.61	0.726	17.91	0.781
Average		**20.74**	**0.856**	18.99	0.851	17.89	0.778	16.13	0.757	12.43	0.521	15.61	0.746	18.49	0.805
Middlebury															
$\beta = 0.75,\ A =$	0.7	14.83	0.791	15.39	0.718	13.47	0.724	16.61	0.836	22.39	0.887	15.80	0.925	23.32	0.883
	0.8	15.82	0.807	16.58	0.738	14.29	0.744	18.11	0.858	21.36	0.873	18.23	0.949	22.18	0.922
	0.9	16.99	0.841	17.01	0.771	15.16	0.769	19.59	0.881	20.05	0.856	21.53	0.963	21.65	0.898
	1	17.84	0.864	17.81	0.761	15.92	0.779	20.58	0.891	18.54	0.831	25.03	0.967	20.16	0.884
Average		16.37	0.825	16.69	0.747	14.71	0.754	18.72	0.866	20.58	0.861	20.14	**0.951**	**21.82**	0.896
$\beta = 1,\ A =$	0.7	14.49	0.748	15.44	0.685	13.17	0.681	16.82	0.807	22.17	0.874	15.76	0.921	21.92	0.892
	0.8	15.82	0.785	16.01	0.673	14.12	0.705	18.75	0.835	20.91	0.863	18.91	0.948	20.02	0.901
	0.9	17.25	0.844	16.58	0.703	15.26	0.754	20.39	0.867	19.19	0.836	22.93	0.961	18.99	0.865
	1	18.31	0.864	17.52	0.752	16.16	0.774	20.61	0.881	16.91	0.793	23.19	0.961	19.14	0.873
Average		16.46	0.811	16.38	0.703	14.67	0.728	19.14	0.847	19.79	0.841	**20.19**	**0.947**	20.01	0.882
$\beta = 1.25,\ A =$	0.7	14.21	0.702	15.17	0.608	12.91	0.631	16.97	0.766	21.94	0.859	15.61	0.899	19.06	0.878
	0.8	15.91	0.771	16.75	0.662	13.98	0.665	19.17	0.799	20.54	0.853	19.01	0.929	17.44	0.845
	0.9	17.63	0.842	17.17	0.667	15.28	0.731	20.63	0.843	18.36	0.814	22.66	0.941	18.87	0.816
	1	18.74	0.862	18.10	0.767	16.28	0.768	19.57	0.859	15.91	0.755	20.09	0.935	17.69	0.833
Average		16.62	0.794	16.79	0.676	14.61	0.698	19.08	0.816	19.18	0.821	**19.34**	**0.926**	18.26	0.843
$\beta = 1.5,\ A =$	0.7	14.04	0.662	15.01	0.607	12.72	0.583	16.98	0.718	21.61	0.841	15.41	0.866	20.04	0.822
	0.8	15.94	0.747	15.94	0.618	13.87	0.619	19.27	0.753	20.19	0.838	18.91	0.897	18.19	0.814
	0.9	18.04	0.836	16.91	0.663	15.36	0.705	20.19	0.812	17.75	0.799	21.27	0.906	16.95	0.791
	1	18.91	0.853	18.35	0.745	16.43	0.771	18.13	0.837	14.73	0.714	17.61	0.897	17.71	0.805
Average		16.73	0.774	16.55	0.658	14.59	0.669	**18.64**	0.78	18.57	0.798	18.31	**0.891**	18.22	0.808

2.5.4 Conclusion

In this paper, we have presented a novel perceptually optimized GAN-based approach toward single image dehazing. Our approach directly learns a nonlinear mapping from the space of hazy images to that of haze-free ones using a deep residue network without estimating transmission maps. By casting the haze-free image as the fixed-point, we can recursively update the residue estimate until the convergence. To ensure visual quality, a discriminative network is introduced for adversarial learning and an adaptive perceptual loss function is developed to handle varying hazy conditions. Moreover, we proposed a novel application of guided filtering into the suppression of halo-like artifacts in dehazed images. Our extensive experimental results have shown that the subjective qualities of dehazed images by our perceptually optimized GAN (POGAN) are often more favorable than those by existing state-of-the-art approaches. The PSNR/SSIM performances of POGAN are also highly competitive especially when the hazy condition varies.

Figure 2.20: Dehazing results on ImageSet A. Left: DCP results [31]. Middle: original images. Right: outputs of POGAN.

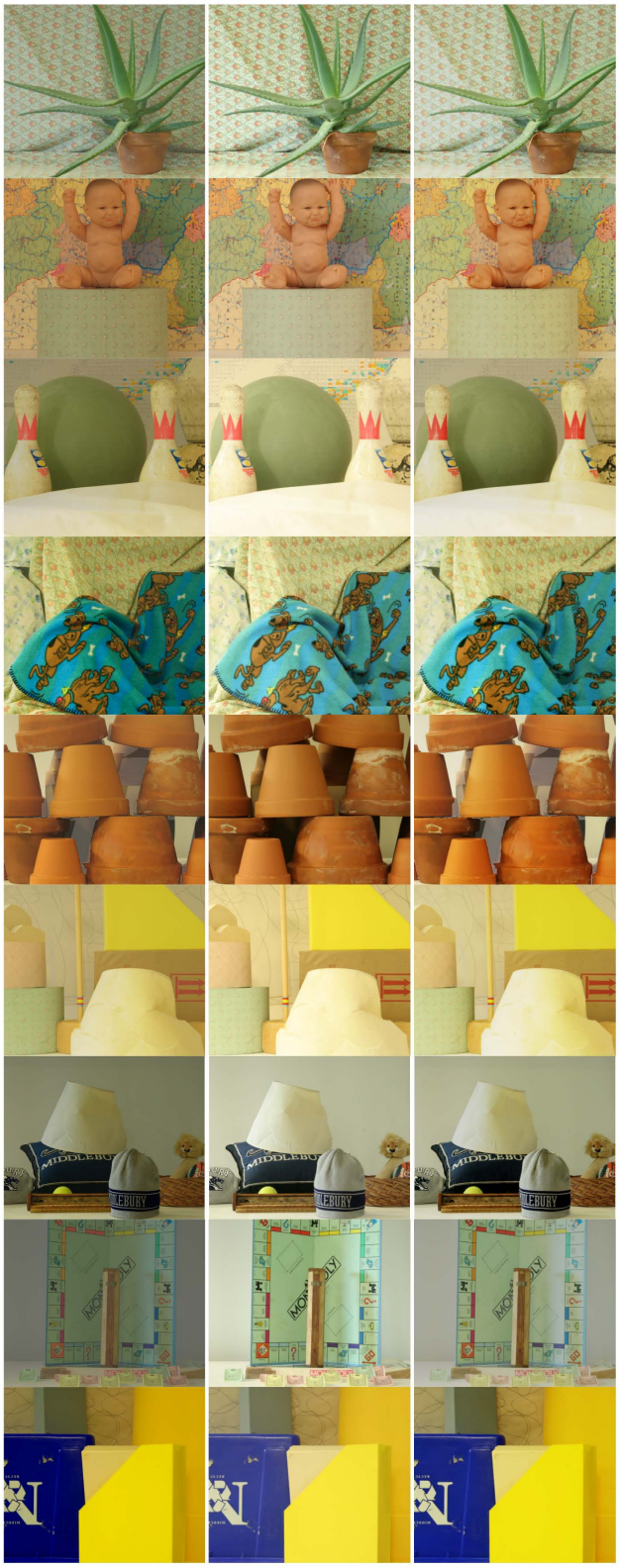
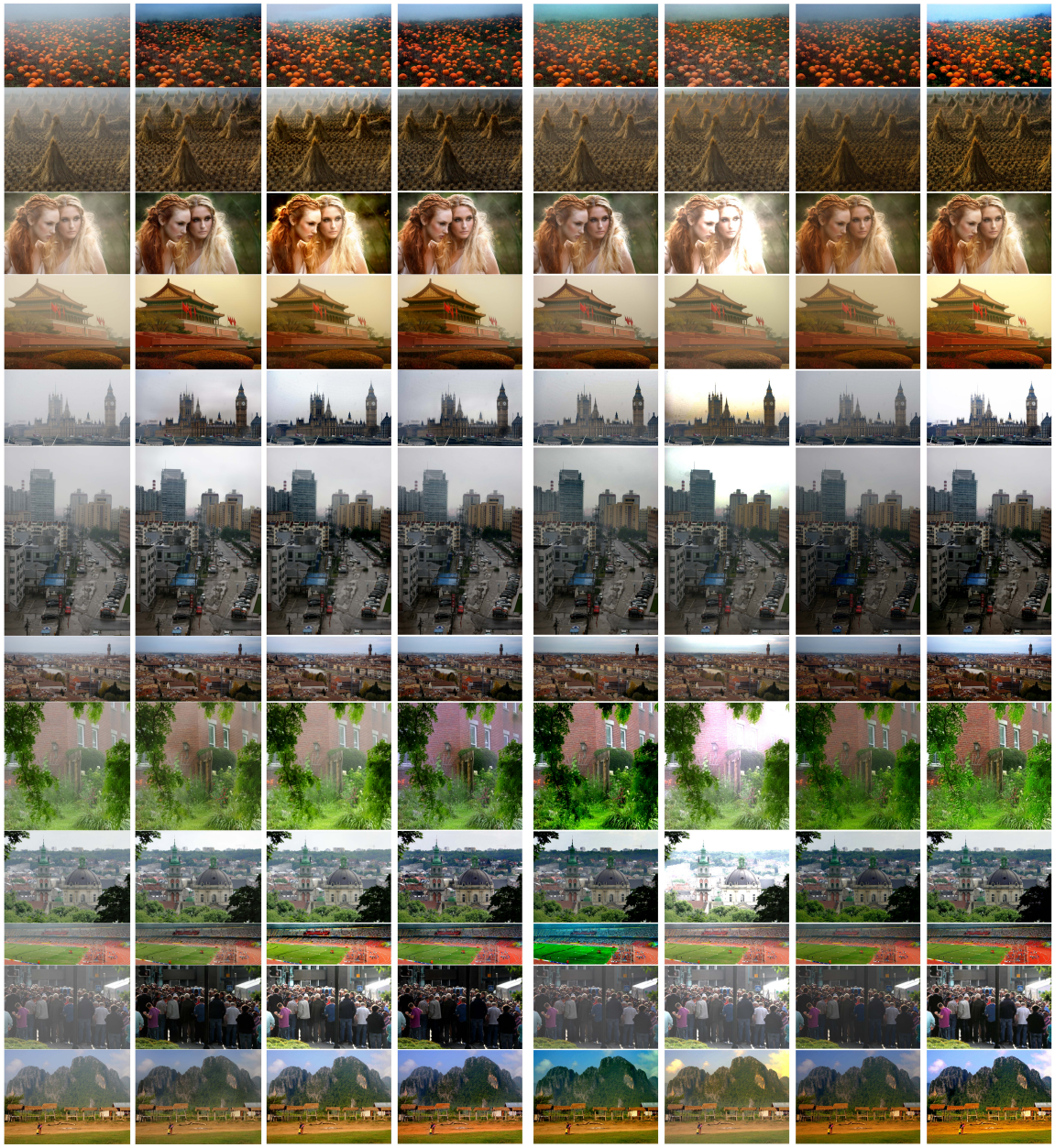


Figure 2.21: Dehazing results on ImageSet B. Left: AOD results [54]. Middle: original images. Right: outputs of POGAN.



(a) Inputs (b) cGAN (c) DCPDN (d) GFN (e) MSCNN (f) DehazeNet (g) AOD (h) POGAN

Figure 2.22: Dehazing results on real-world images.

Chapter 3

CNN-based Loop Filter and Super-Resolution in HEVC

3.1 CNN-based Loop Filters

Loop filtering aims at removing compression artifacts and improving visual quality of a video. It is an important component in High Efficiency Video Coding (HEVC). Traditional loop filters in HEVC such as deblocking filter has limited performance with respect to filtering accuracy. Inspired by the success of deep learning, some Convolutional Neural Network based loop filters have been proposed to address the issue of traditional filters. Follow this line of thought, we propose a deeper neural network filter for further coding efficiency improvement. When compared with traditional filters, the proposed approach could fulfill multiple filtering tasks with only one single model. It is also better when compared with current CNN-based loop filters with respect to BD-rate savings. The rest of this Section will review previous loop filters including both traditional model-based and CNN-based approaches, propose our new model, and report experimental results.

Figure 3.1: An example frame illustrating the compression artifact. The original frame is on the left, and compressed frame is on the right. Notice the blocking and blurry effect on the right, especially on the face and the texture of the suit.

3.1.1 HEVC Loop Filters

The current state-of-the-art video coding standard is called HEVC or H.265 which is the successor of H.264. Compared with H.264, HEVC could reduce 50% bit-rate while remain the same image quality. HEVC belongs to lossy compression, thus, the resulting image contains some noticeable artifacts especially under low bit-rate. For example, blocked-based compression mechanism often yields blocking artifacts, and these artifacts are located on the border of blocks. Image blurriness is another type of artifacts due to high frequency information loss during the compression. Figure 3.1 is an example illustrating the compression artifact, notice that after compression (as shown on the right-hand side of Figure 3.1), there is significant quality degradation caused by blocking and blurry artifacts. In summary, the quality of a video is affected by compression artifacts. How to remove these artifacts is an important and widely studied problem.

There are two artifact removing filters (also called in-loop filters) in HEVC, including deblocking filter (DF) and Sample Adaptive Offset (SAO) as shown in Figure 3.2. The

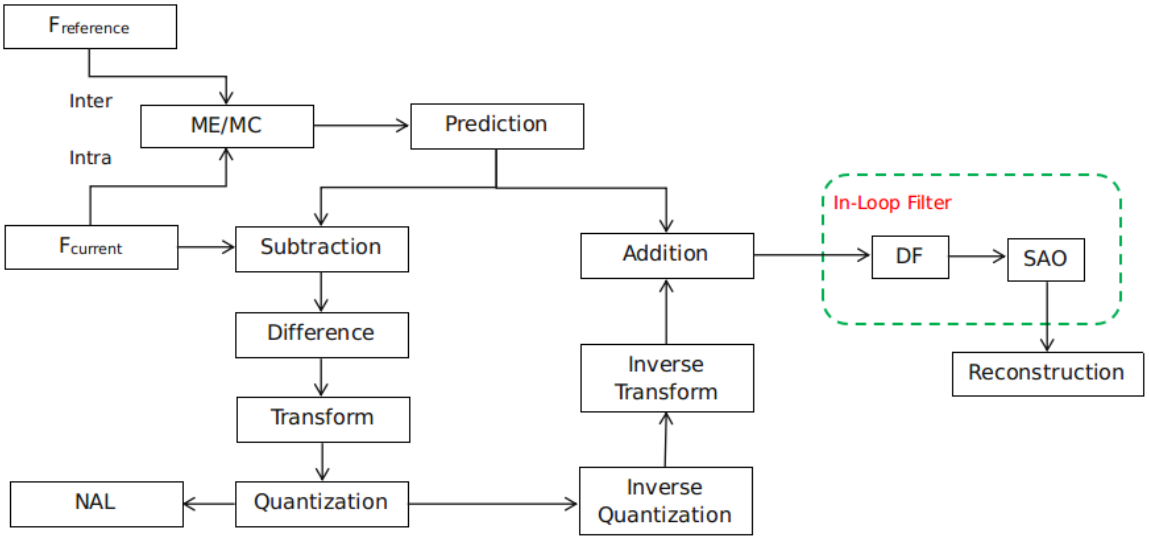


Figure 3.2: Illustration of loop filter in HEVC as shown in highlighted area.

former one is designed to remove blocking artifacts along the boundary of blocks, the later one reduces sample distortion [25]. Using DF and SAO together could potentially reduce bit-rate and improve both subjective and objective quality. Subjective quality refers to the visual quality, objective quality is measured by common performance metrics such as Pixel-to-Noise-Ratio (PSNR).

The next generation video coding standard is called versatile video coding (H.266). Researchers has proposed two more filters including bilateral filter (BF) and adaptive loop filter (ALF). They have shown that using the four filters mentioned previously could produce better results.

3.1.2 Related Work

Deep learning is one of the most powerful tools in the field of computer vision. Computer vision aims at using machine to solve vision-related problems, such as image denoising, image super-resolution, and object detection/recognition. Loop filtering belongs to computer vision as well. Deep learning refers to building a network which is composed of large number of neurons, mimicking brain neural system of humans. Each neuron contains weight

and bias which are used to convolve with input, produce feature maps, and generate output at the end of the network. Most of the current state-of-the-art computer vision approaches are deep learning based, thus, deep learning has been extended to be implemented in many new areas, including loop filtering in video coding. First, loop filtering aims at removing artifacts in an image/video, this problem can be described using the following equation:

$$y = x + a, \tag{3.1}$$

where x stands for the original image, y refers to compressed image, a is the additive artifacts, a is also called residual. In machine learning, recovering the residual between input and output is called Deep Residual Learning (DRL). DRL could compute the artifacts between a given compressed image and restore a clean image. In the following, we review previous residual learning based loop filtering. Park and Kim [75] proposed an entry-level residual learning model to replace SAO in HEVC. The model has a very simple architecture including two layers. They are of size 6433 and 3255 respectively ($c \times k \times k$), where c refers to the number of feature maps in each layer, and k is the size of each convolutional kernel. There is also a summation layer at the end of the network in order to compute the output. Based on this architecture, they have trained two models using Quantization Parameter (QP) ranges from [22,29] and [30,39], respectively. The model selection in testing phase is based on the QP settings. The author reported 2% BD-rate saving over HEVC where BD-rate is a trade-off between distortion and bit rate.

Dai, Liu and Wu [16] proposed a four layer fully convolutional model. When compared with [75], the main difference is the concatenation of feature maps that are computed using different size of convolutional kernels such as (55, 33, 11). This allows obtaining a more diverse feature maps to better describe the related high frequency information. At the same time, [16] used more parameters since their network is wider than [75]. The author has trained four models under QP settings of 22, 27, 32, and 37, to replace both SAO and DF

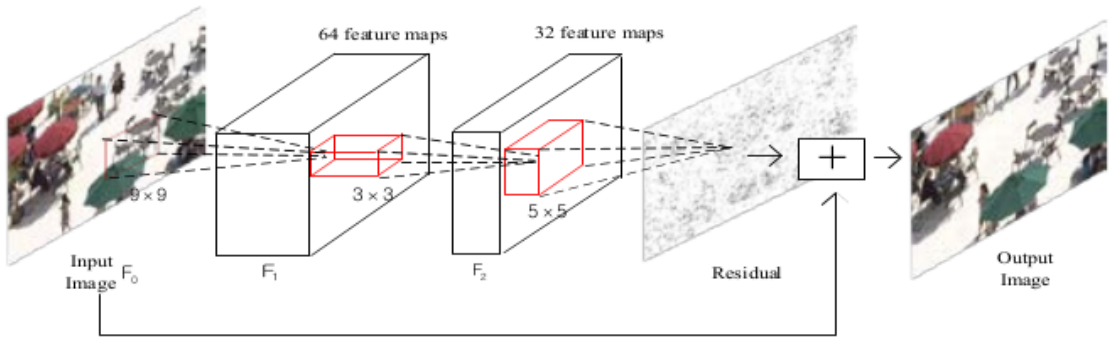


Figure 3.3: CNN architecture from Park and Kim [75].

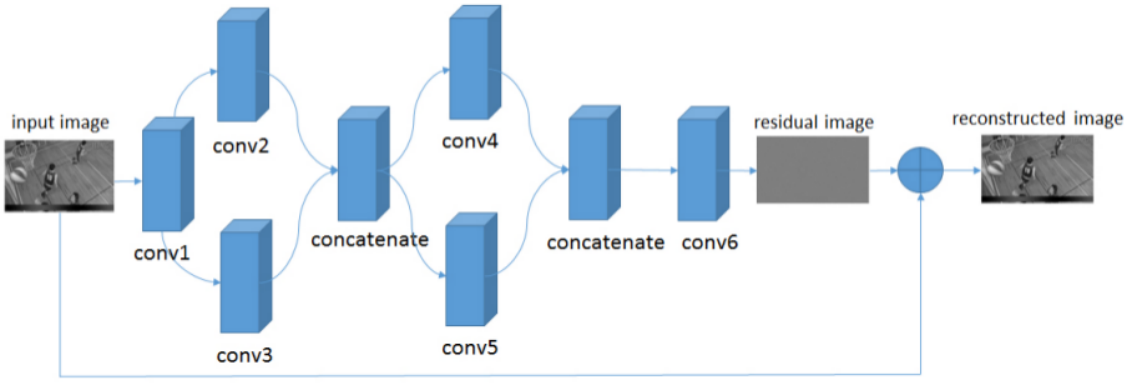


Figure 3.4: CNN architecture from Dai, Liu, and Wu [16].

in HECV. They have claimed 4% BD-rate saving.

In Yang *et al*. [107], the authors proposed a CNN network by fusing Inter and Intra modes together. They have reported improved performance on both HEVC I and P/B frames. Li *et al*. [56] presented a fully connected neural network for Intra prediction, around 4.5% bitrate saving is reported but the overall network size is very large due to the fully connected architecture. He *et al*. [34] incorporated the coding unit (CU) size information into the CNN which guide the quality enhancement process. They have reported improved performance compared with the non-guidance ones.

Table 3.1: Parameter settings in Dai, Liu, and Wu [16].

Layer	Layer 1	Layer 2		Layer 3		Layer 4
Conv. module	conv1	conv2	conv3	conv4	conv5	conv6
Filter size	5×5	5×5	3×3	3×3	1×1	3×3
# filters	64	16	32	16	32	1
# parameters	1600	25600	18432	6912	1536	432
Total parameters	54512					

3.1.3 The Proposed Approach and Results

It can be seen that [75] and [16] have their own limitations. First of all, they can only replace one or two traditional filters (DF and SAO), while there are two more filters in the newly established video coding standard (BF and ALF). Second, they train multiple CNN models toward different QP or different range of QP. Third, the best they can do as compared to baseline approach is 4% gain, there should be enough room we can further improve the performance. Based on these findings, we propose to train a new CNN model with the following expectations:

- The new model should be able to replace all of the four filters, including BF, DF, ALF and SAO.
- The new model should be versatile enough to deal with multiple QPs with one single model.
- The new model should achieve more than 4% BD-rate savings, which surpasses current approach.

Based on those expectations, we propose a deep residual model, as shown in Table 3.2. The new model is composed of 8 layers, the first layer is data layer, the second to 6th layer is ConvR layer, which stands for convolution and rectified linear unit. The 7th layer is summation layer which adds the residual to the input data. The last layer computes the loss

Table 3.2: The architecture of the new model. ConvR stands for Convolution and Relu.

	Layer 1	Layer 2	Layer 3	Layer 4	...	Layer n-2	Layer n-1	Layer n
Name	Data	ConvR 1	ConvR 2	ConvR 3		ConvR n-3	Sum	Loss

during the training process.

Following are some training details: first we turn off all traditional filters in JEM 7.1, to generate the reconstruction. We then use the reconstruction to produce some Y/U/V patches, and these patches are used as input for our network. The experiment is conducted in Caffe, with Intel i7-6850k CPU, and NVIDIA 1080ti GPU. It took around 24 hours for our model to converge. All other settings, such as solver, learning rate, are all set to defaults.

We use common testing sequences for video coding, such as BQSquare, BQMall, BasketballDrill. For every testing sequence, we encode it using JEM 7.1, with four traditional filters turned on, record the bit rate and Y/U/V PSNR, as the ground truth. Then, we use our CNN model to replace four traditional filters and record the results. With before and after bit rate and PSNR available, we compute BD-rate savings. Table 3.3 shows the performance, we observed around 6% BD-rate saving. Figure 3.5 shows the comparison of visual quality: the left and right columns shows the before and after results. It can be noticed that with our CNN model, the blocking and blurry artifacts can be suppressed after the filtering process.

3.1.4 Conclusion and Future Research

This chapter introduces HEVC in-loop filter, reviews traditional loop filters and CNN-based filters. On top of this, we proposed a new CNN model which has deep residual learning structure. The new model is deeper, and it has more feature maps in each layer. Objectively, it achieved 6% BD-rate savings, and objectively, the new approach could improve the visual quality. There are two directions regarding future research: First, as extending

Figure 3.5: Visual quality comparison before and after using the proposed CNN filter.

network depth/width does not improve the result, how to further implement novel network architecture in order to have larger gain is our focus. Second, studies show that super-resolution could improve coding performance under low bit rates, and the current state-of-the-art super-resolution approaches are based on deep learning. It is desirable to merge the tasks of super-resolution and loop filter in one single neural network, which is the direction we want to discover.

3.2 Content Weighted CNN Loop Filter

3.2.1 Introduction

The content in an image/video differs significantly: such as edges vs textures, human faces vs bodies, foregrounds vs backgrounds, etc. Different content may require different compression rate, which is controlled by QP settings. For example, people tend to pay more attention to human face rather than the body and environmental background in a portrait photo. Thus, it is reasonable to use lower QPs in face area, and higher QPs in the rest of the photo. Doing so has two advantages: it ensures the visual quality of facial area, it also reduces bit-rate as higher QPs are used in non-face area. We take the analogy from compression to loop filtering, and we propose a content weighted loop filter whose filtering strength is controlled by the content of the region. In more important areas such as face and edges, our model increase filtering strength. In less important areas including textures and backgrounds, the filter becomes gentler. Different brightness on the right-hand side stand for different weights in the filtering process, and this information is then utilized by our content weighted model to control the filtering strength. This content adaptive/weighted filtering approach could potentially yield more gains compared with previous approaches.

3.2.2 The Proposed Content-Weighted approach

Our network is composed of two networks: The Content Weight Network (CWN) and The Deep Residual Network (DRN), as shown in Figure 3.6. An input image will go through four steps to produce the final output:

Step 1. The CWN computes the weight of each region, more important regions tend to have higher weight than less important regions.

Step 2. The DRN has similar structure as in [16] and [75]. It includes Convolutional layers and ReLu layers. The Convolutional layer extracts feature from the input, which serves as input of next layer. ReLu stands for rectified linear unit, it performs max(0, x) operation. ReLu is one of the non-linear activation functions to speed up the training and testing of neural networks.

Step 3. The resulting feature maps from DRN and CWN are element-wise multiplied together, and feed into another Convolutional layer which simply outputs one feature map (the residual component in Equation 1.).

Step 4. Finally, the residual is added to the input through element-wise sum, computing the output.

3.2.3 Variations of the Proposed Approach

Depending on the desired model complexity, there are some variations of the proposed approach. The variation is mainly related with network complexity in the Deep Residual Network. There are two factors deciding the network complexity: the network depth and the number of feature maps in each layer. Regarding the depth, a simple model could have 2-6 layers as the ones in [16] and [75], a larger model could have around 12-14 Convolutional layers. The number of feature maps in each layer could range from 16 to 512. These are the two factors that will bring variations to the proposed approach. Content weight network

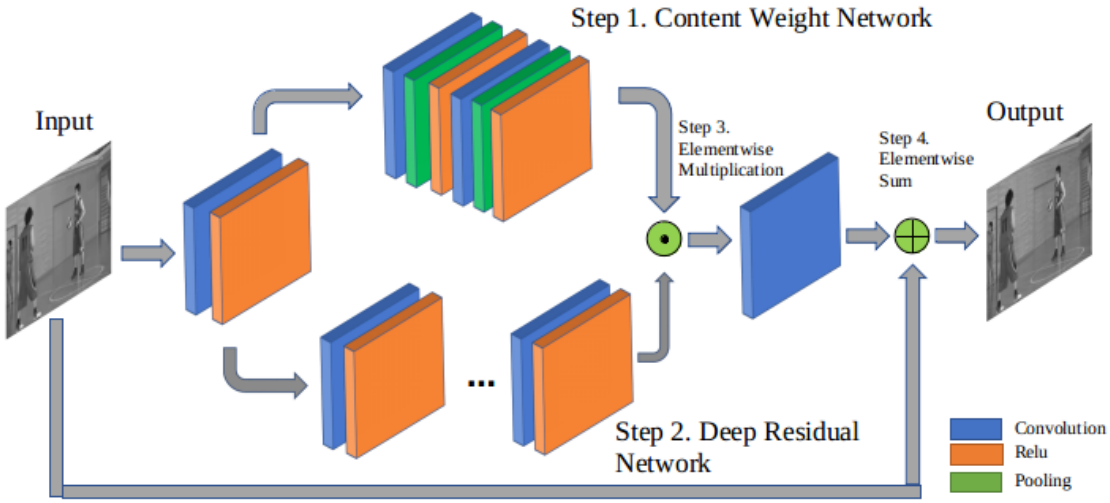


Figure 3.6: Architecture of the proposed model. There are two major components: the content weight network (top) and deep residual network (bottom).

can use encoding information, e.g., quantization parameters, motion vectors, coding unit partition information, and so on, as one of the input information. In other word, the content weight is not only derived by pixel information but also by the encoding information.

3.2.4 Implementation Details

Content Weight Training Data

In order to train a content weight network, we propose to use the adaptive QP mode in X264 compression and obtain an adaptive QP map for the purpose of training the network. The detailed steps are listed below:

Step 1: We use X264 to compress all training images with adaptive QP flag turned on, producing all associated .264 files.

Step 2: We use JM 19.0 to decode all the .264 files generated in last step, at the same time, turn on the QPMap flag, and obtain an adaptive QP map.

Step 3: We normalize the obtained QP map, with the QP range in [22, 41].

Figure 3.7 shows the content weight map obtained using the above steps.

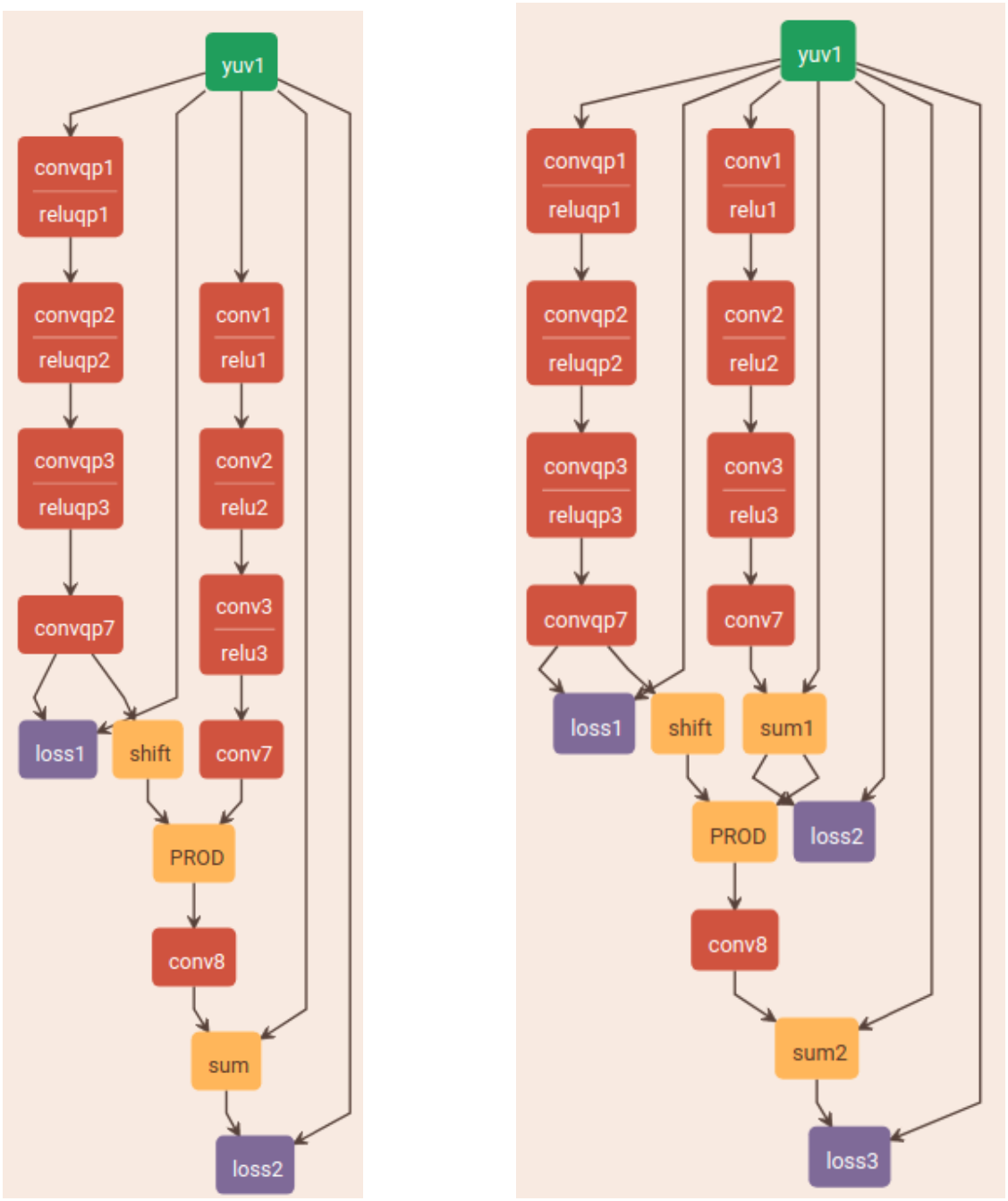
Figure 3.7: Content weight of an example image. Different brightness on the right-hand side stand for different weights in the filtering process, and this information is then utilized by our content weighted model to control the filtering strength.

Network Structure

Figure 3.8a shows the initial network structure of the CNN. It includes two components, a content weight network on the left and a residual network on the right. There are two loss functions. The first loss function computes errors between content weight image input and output, the output is shifted by some constant afterwards. The residual network on the right computes the residuals, and the output is elemental wise multiplied with content weight. In the end, loss 2 computes the error from the enlarged residual and original label.

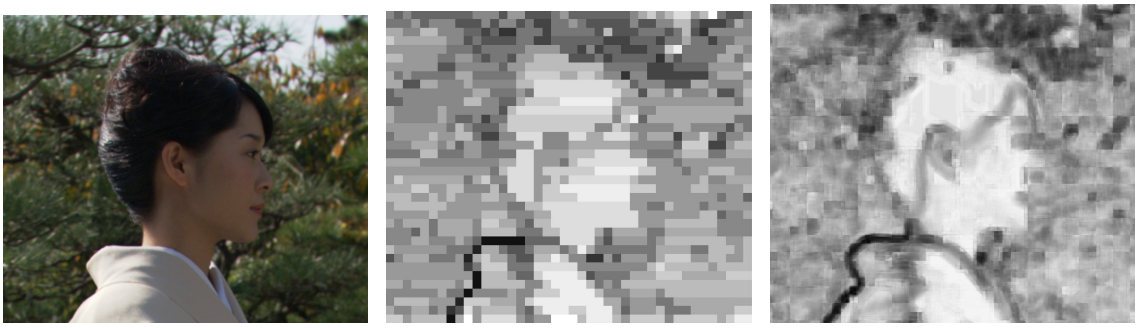
Results and Discussions

The output of the content weight network is shown in Figure 9. On the left is the original image, the output of X264 is shown in the middle, and content weight output from CNN is shown on the right. It can be seen that CNN output is very similar as X264 output, which provides good foundation for us to move forward.



(a) (b)

Figure 3.8: Left: The initial network architecture of the proposed approach. Right: The updated architecture with another loss function added to train each part of the network independently.



(a) Input (b) X264 (c) CNN

Figure 3.9: The output of the content weight network. On the left is the original image, the output of X264 is shown in the middle, and content weight output from CNN is shown on the right.

Figure 3.9. The output of the content weight network. On the left is the original image, the output of X264 is shown in the middle, and content weight output from CNN is shown on the right. We have obtained 34.1 average PSNR when only using residual network. The best performance of content-weighted residual network is just the same, 34.1, in terms of PSNR. It means we still need to improve the network. The reason is that the original network only has two loss layers, as a result, part 1 and 2 interacts with each other, which does not comply with our initial intention. We want part 1 and part 2 to be isolated from each other so that we can achieve the enlarged residual and see it helps improve the result. So, the network is modified as shown in the Figure 3.8b: we add another loss function to the network and train each part of the network independently. By doing so, we are expecting to see gains over the residual only network.

3.3 Super-Resolution under Low Bandwidth in HEVC

There exists the following scenario: when bandwidth is really limited, the compression must use very large QP, as a result, the quality of reconstruction is very low. For example, the following figure shows when bandwidth is 1793.2 kbps, QP 41, the original v.s. reconstruction. The reconstruction is mostly blurred.

Figure 3.10: Original v.s. reconstruction when bit rate is 1793.2 kbps, QP 41.

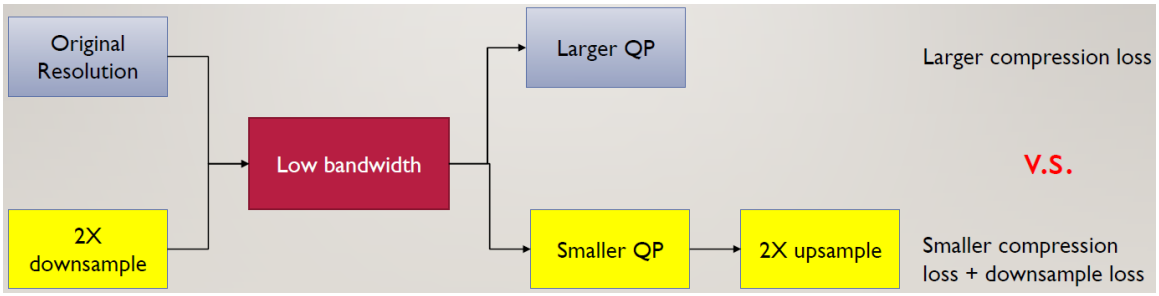


Figure 3.11: The usage of super-resolution as an alternative when bandwidth is very low.

There exists an alternative approach: the use of super-resolution in HEVC. One could down-sample the video at the beginning, use a smaller QP to compress the video, and finally do an up-sampling. The counter-part is the alternative approach will produce smaller compression loss and down sample loss, the original approach will have larger compression loss. So eventually we are comparing which loss is smaller between the two.

With respect to up sampling, one can use bicubic/bilinear, or CNN. The advantage of using CNN as oppose to using bicubic/bilinear operator is that CNN could handle both down sample loss and compression loss simultaneously. This could potentially yield more gains. Figure 3.12 shows some preliminary results toward CNN-based super-resolution. On the left is the bicubic result with QP 33, in the middle is full resolution result with QP 41, and on the right is CNN result using QP 33. Its obvious that CNN result is a little bit better. In this Section, a multi-scale super-resolution module will be presented in order to improve image quality in HEVC under low bandwidth.

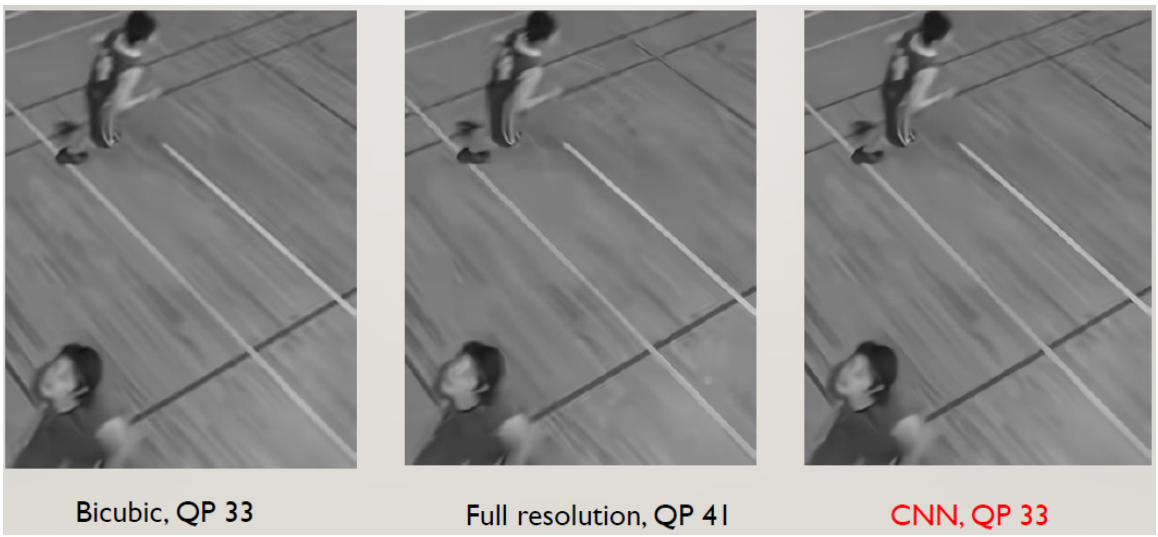


Figure 3.12: Comparing the visual quality of reconstructions obtained by bicubic upsampling, full resolution, and CNN-based super-resolution.

3.3.1 Related Work on Single Image Super-Resolution

Single image super-resolution (SR) [40] is a classical computer vision problem which aims at increasing the resolution of an image given an input low resolution image [18]. Traditional example-based approaches exploit internal global and local similarities of an image [23, 27], or try to learn a pair of low-to-high resolution exemplar from an external image [8, 12, 24]. Recent state-of-the-art super-resolution approaches achieves much better performance with Convolutional Neural Networks, such as SRCNN [18], VDSR [47], ESPCN [87], RCAN [112]. We will review both example-based and CNN-based SR approaches in the following and compare their main contributions in detail.

Example-based SR Approach

[24] proposed an approach called VISTA-Vision by Image/Scene Training which is applied to the SR problem. They generate synthetic world of scenes and corresponding rendered images with a Markov Network. The network is then solved with Bayesian belief propagation in order to find the MAP (maximum a posterior probability of the scene). [12] is inspired by locally linear embedding based on the assumption that low and high resolution

image patches shares similar local geometry in their feature spaces, respectively. [106] approaches the SR problem from the compressed sensing perspective that the sparse representation to an dictionary can be used to recover a high-resolution image. [27] proposed a unified framework to combine both multi-image SR and example-based SR. The key insight is that most of the natural images has a lot of redundant patches in either the same scale or different scales. Patches with similar scales is the input for multi-image SR approaches, and patches with different scales drives the SR with example-based approaches. The author fused these two by considering both similar and different scale of redundant patches in a natural image. [103] proposed a sparse signal recovery approach based on the finding that an image could be represented as a sparse linear combination of patches from an over-complete dictionary. The author computes the sparse representation of each input low resolution patch, the coefficient of the representation is then used for generating SR images. It is accomplished by training a low and high resolution image patches dictionary, and find the correspondence between the two dictionaries. In [23], the author used local-self patches as the main SR input instead of external database. The benefit is that searching the similar patch locally is way more faster than searching in an external database. They reported their algorithm is simple, efficient, and can be implemented on a GPU. Meanwhile, high-quality resolution enhancement to both image and video has been demonstrated.

An SR method based on non-negative neighbor embedding has been proposed in [8] which belongs to the example-based SR family. It trains a dictionary with pairs of low and high resolution patches. The dictionary is then used to search for the corresponding high resolution patch of the input low resolution one. The input low resolution image is represented as a weighted combination of K nearest neighbors in the dictionary with the assumption that the output high resolution image preserves the same feature vector. [105] presented a coupled dictionary training method for image SR. Their work is also based on the idea that the sparse representation of both low and high resolution image patch are similar, in other works, the high resolution image patch can be reconstructed based

on the sparse representation of it's corresponding low resolution patch. The optimization is solved by L1 norm minimization. [108] improved the SR performance by introducing some modifications to the local sparse land model proposed by [106]. The main modification is done toward computational complexity reduction and algorithm architecture improvement.

[102] proposed to split the feature space into different subspaces during the process of reconstructing high resolution image. Effective mapping functions are created by learning priors for each subspace. [104] also exploited the local self-similarity of a given image patch, the so-called in-place example. Then, a first-order approximation of nonlinear mapping function is learned to reconstruct high resolution image patches. [41] extended the similar idea of dictionary learning based on image transformation. The transformation is accomplished by local parametric regression of sparse feature representations. The training image patches are transformed into retrievable local clusters which is used for fast indexing of a testing image.

Learning-based SR Approach

SRCNN [18] is one of the learning-based SR approaches that fully adopt a deep convolutional neural network to take the low resolution input and output a high resolution one. There are three components include: patch extraction and representation, non-linear mapping and reconstruction. Mean squared error (MSE) is used as the loss function which computes the distances between the reconstructed image $F(Y;\theta)$ and ground truth high resolution image X. The network consists of two layers with 64 and 32 feature maps, respectively, the author claimed state-of-the-art performance compared against traditional example-based approach. Following the aforementioned work, FSRCNN [20] is proposed to increase the processing speed of SRCNN by introducing deconvolutional layers and adopting smaller filter sizes and more mapping layers. As a result, the author claimed 40 times speed with even better restoration quality. Kim *et al*.[48] proposed a deeply-recursive convolutional network (DRCN) with 16 recursion layers. Skip connection is introduced to

ease the problem of vanishing gradients. Kim *et al*.[47] introduced a very deep convolutional networks for SR (VDSR). The network takes interpolated high resolution image as input, followed by 20 layers of $Conv + Relu$, and one skip connection for element-wise summation of residual and initial interpolated image. Since this work, residual learning has been more and more popular in single image SR.

Lim *et al*. [58] proposed an enhanced deep residual network for SR (EDSR). The main contribution of this paper is the introduction of repeated residual blocks ($ResBlock$) without batch normalization, and the traditional deconvolution upsampling layer is replaced by pixel shuffle layer which could achieve any arbitrary number of upsampling ratio. [51] proposed a Laplacian Pyramid Super-Resolution Network (LapSRN) to progressively reconstruct high resolution image with multiple steps, each step with a factor of 2. For example, an $8x$ SR image is produced by $2x$, then $4x$, then $8x$, respectively.

[52] is another milestone in single image SR by introducing a photo-realistic Generative Adversarial Network (GAN). Beside the generative module which consist of recursive residual learning network, there is a discriminative network which classifies the output of generative module. The author claimed that even though they did not achieve state-of-the-art in terms of PSNR, but the visually quality of the reconstructed high resolution image is way more superior thanks to the perceptual loss.

Most recently, a multi-scale residual network (MSRB) [55] is proposed with concatenation of features from multi-scale filters, or the so-called multi-scale residual block. The output of each block is fused hierarchically for global features, followed by the reconstruction module with 1×1 convolutions to output the high-resolution image. [35] proposed a very similar idea by fusing the feature maps from each multi-scale residual block. The main difference is that the residual block is slightly different from [55]. Zhang *et al*. [112] proposed a residual channel attention network (RCAN) to deal with abundant low-frequency information. RCAN features a global long skip connection and a local short skip connection. The author also introduced adaptive rescaling channel-wise features by interdependencies.

Residual dense network (RND) [113] and enhanced SRGAN (ESRGAN) [97] are proposed in 2018 which emphasize on objective and subjective quality, respectively. Figure 3.13 and Table 3.4 compare previous works on single image super-resolution.

3.3.2 The Proposed Multi-Scale Super-Resolution Approach

Quantization Parameter and Rate Control

In H.264, each block of a current frame is predicted with motion estimation and motion compensation as shown in Figure 3.14. The prediction is obtained using either Inter (temporal) or Intra (spatial) mode. Then, the difference (residual) is computed, transformed and quantized, followed by reordering and entropy coding before passing to a network abstraction layer (NAL) for transmission or storage [45]. Here we focus on the quantization process and discuss it's effect toward rate control.

Let $\mathbf{E}$ represent a 4×4 block of predicted and transformed residuals. The approximated discrete cosine transform $\mathbf{Y}$ can be computed by [96]

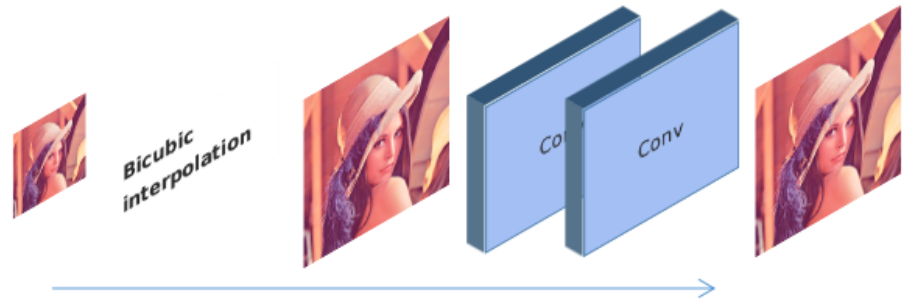
$$\mathbf{Y} = \mathbf{Z} \odot \mathbf{S}, \quad \mathbf{Z} = \mathbf{TET}^T \tag{3.2}$$

where $\odot$ denotes element-wise multiplication. $\mathbf{T}$ is the transform operation and $\mathbf{S}$ is the post-scaling operation defined in [81].

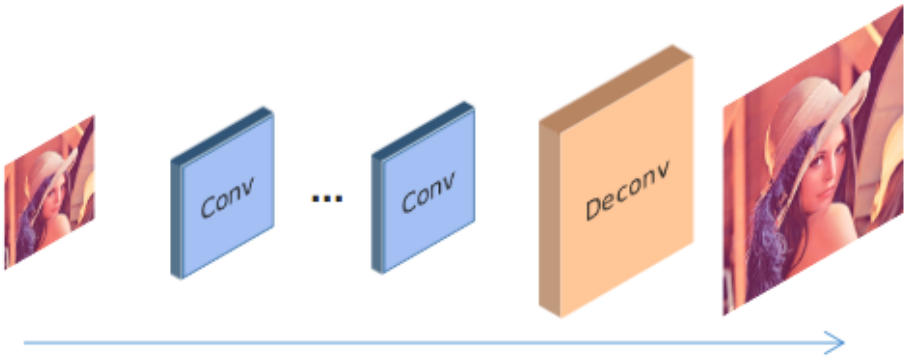
The value of quantization coefficient Y_j can be computed by [96]:

$$\hat{Y_j} = I_j \times q = \text{sign}(Y_j) \left\lfloor \frac{|Y_j|}{q} + 1 - \alpha \right\rfloor \times q \tag{3.3}$$

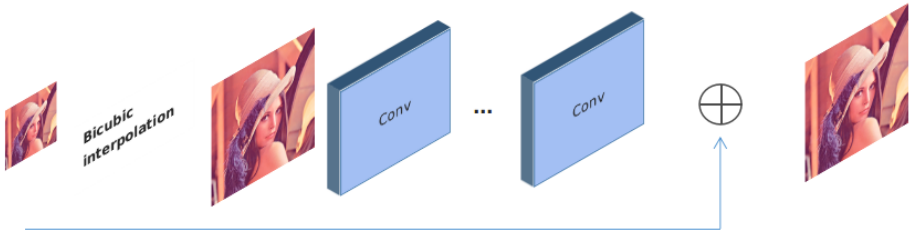
where q is the quantization step, α is typically selected as 2/3 for Intra blocks and 5/6 for Inter blocks which controls the width of the dead zoon around zero. I_j stands for the quantization index to be transmitted. The quantization step q can be computed using the



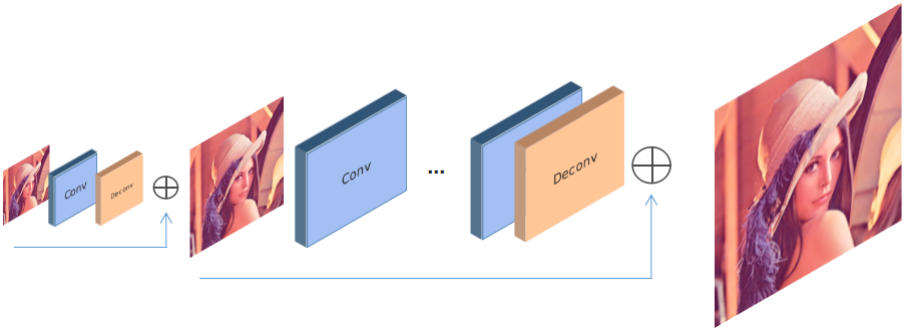
(a) SRCNN



(b) FSRCNN



(c) VDSR



(d) LapSRN

Figure 3.13: Compare different architecture of SRCNN [18], FSRCNN [20], VDSR [47] and LapSRN[51].

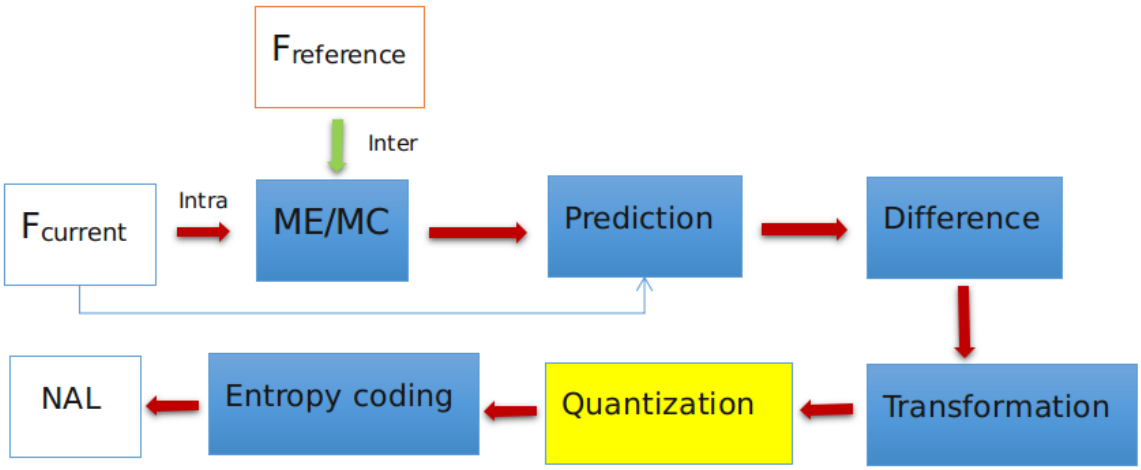


Figure 3.14: Overview of H.264 encoder.

following equation:

$$q = q_B \ (\mathrm{mod}\ (QP, 6))2^{\lfloor QP/6 \rfloor} \tag{3.4}$$

where q_B is defined in [81]. From the above equations we can see that large QP values will result in large quantization step, which crudely estimate the spatial transform and resulting in few coefficient. In contrast, small QP value means refined and more accurate approximation of the block's spatial transform, at the cost of more bits.

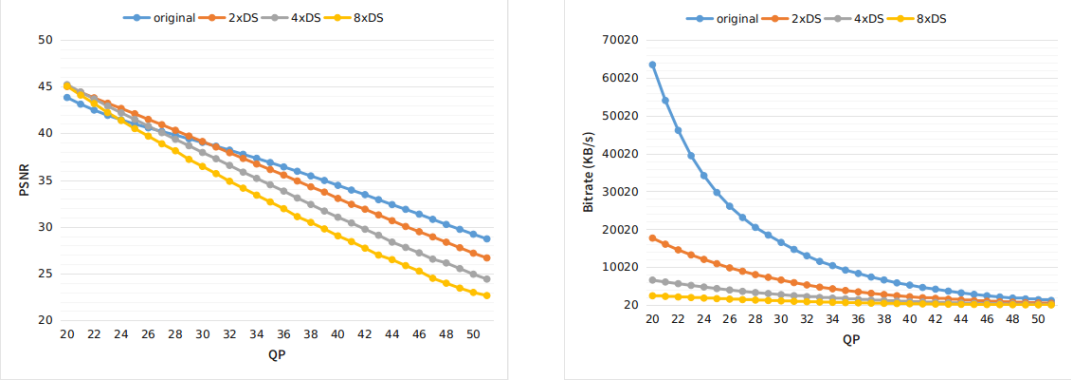
In H.264, the QP value is predefined based on channel bandwidth as follows [15]:

$$QP = \begin{cases} 40 & bpp \leq l_1, \\ 30 & l_1 < bpp \leq l_2, \\ 20 & l_2 < bpp \leq l_3, \\ 10 & bpp \geq l_3 \end{cases} \tag{3.5}$$

where

$$bpp = \frac{R}{f \times N_{pixel}} \tag{3.6}$$

R is the target rate, f is the number of frames, and N_{pixel} is the number of pixel in a picture.



(a) QP v.s. PSNR

(b) QP v.s. Bitrate

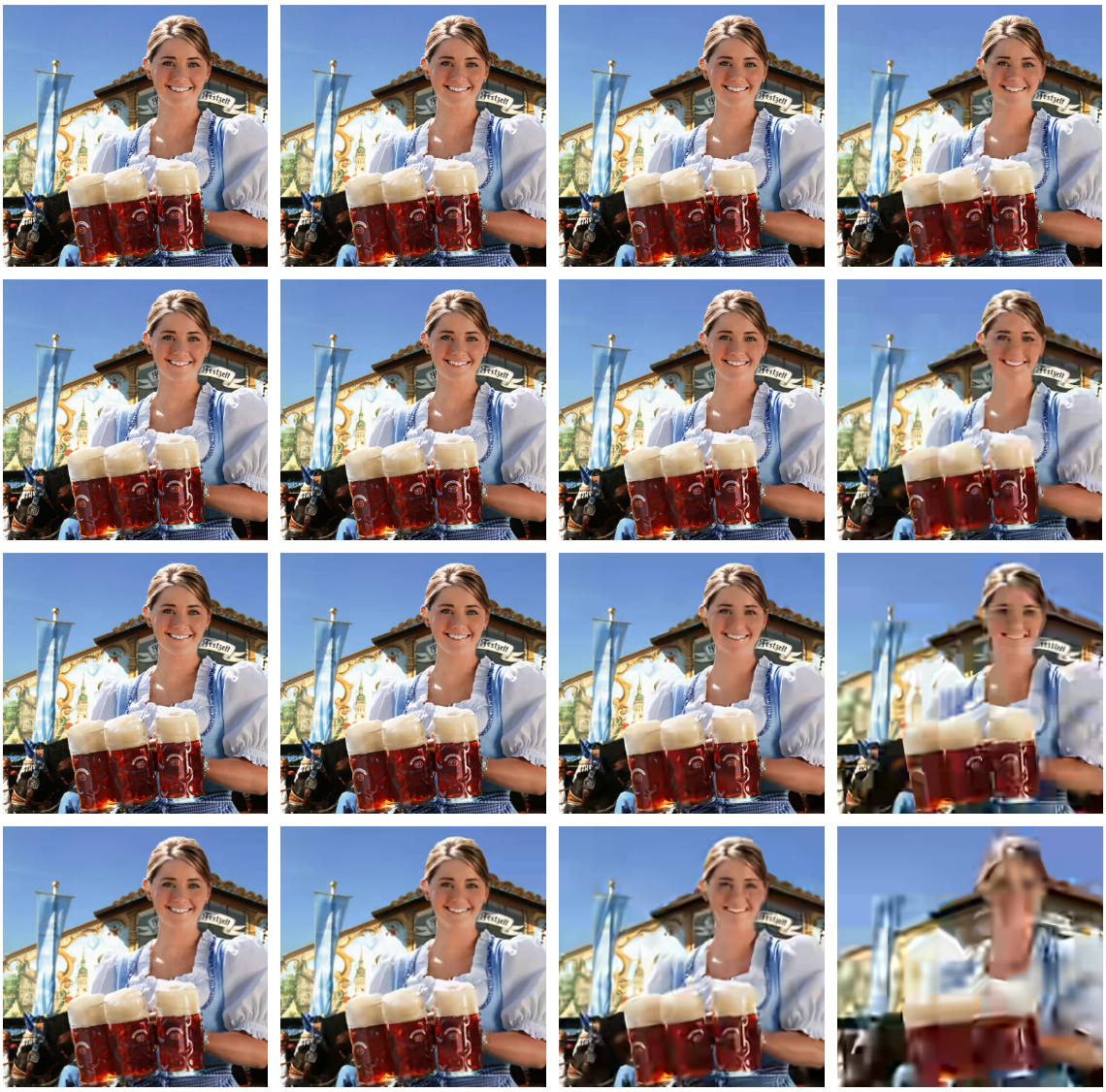
Figure 3.15: (a) compares QP versus PSNR with original resolution, 1/2 resolution, 1/4 and 1/8 resolution respectively. (b) compares QP versus bitrate. All results are obtained by encoding a 2016×1984 resolution video with one frame in JEM 7.0.

Figure 3.15a compares QP versus PSNR with original resolution, 1/2 resolution, 1/4 and 1/8 resolution respectively. Figure 3.15b compares QP versus bitrate. All results are obtained by encoding a 2016×1984 resolution video with one frame in JEM 7.0. It can be observed that the larger the QP, the smaller the PSNR. Meanwhile, under the same bitrate, one can encode a lower resolution video sequence with smaller QP.

Figure 3.16 shows the visual comparison of reconstruction generated in encoding process using JEM 7.0 with different QP settings. The resolution of the frame from top to bottom row are 2016×1984, 1008×992, 504×496, and 252×248, respectively. We can observe from the figure that the visual quality of 1/4 resolution, 1/2 resolution, and original resolution looks fine under some QP settings. However, the 1/8 resolution frame (last row) appears to be blurry due to too much high frequency information loss in the initial bicubic downsampling process.

The Deeper, the Better in Video Coding?

As we can see from Table 3.4, there are various learning-based SR approaches, the most obvious pattern one can infer from the table is that the number of CNN layers is increasing in chronological order: In 2014, SRCNN [18] has 3 layers, each of which has 64 feature



(a) QP 20 (b) QP 30 (c) QP 40 (d) QP 50

Figure 3.16: Visual comparison of reconstruction generated in encoding process using JEM 7.0 with different QP settings. The resolution of the frame from top to bottom row are 2016×1984, 1008×992, 504×496, and 252×248, respectively.

maps, while in 2018, RCAN [112] has 400 layers. The model size also increased from 32.4KB to 59.88MB as shown in Table 3.4.

In VDSR [47], the author argued that as the number of layers increase, SR performance improves significantly. They have shown the PSNR performance of networks ranging from 5 to 20 layers, and the larger the depth, the better the result. This is true only under the circumstance that the amount of resources is unlimited, such as memories and GPU. But in our case when bandwidth is limited, it is essential to keep the size of network small. Because the CNN network weights will require extra bits to transmit and store, if the network itself is very large, it is impossible to win the bitrate v.s. PSNR trade-off.

The Network Architecture

As shown in Figure 3.16, the visual quality of one half and one quarter resolution under small QP value rivals the the original resolution under large QP value. Thus, we propose a Convolutional Neural Network to fulfill the SR task in HEVC under low bandwidth. Traditional SR networks takes a bicubic downsampled image as input, the downsampling factor is typically within the range of $[2, 16]$, followed by either bicubic pre-upsamling, or deconvolution, or pixel shuffle to bring the input image back to the original size in a direct or progressive manner. Direct approach enlarges the image to the desired size in one step, while progressive approach use intermediate upsampling factors to achieve the desired resolution. Then, some residual learning steps are fulfilled by convolutional layers and rectified linear unit (ReLu) with skip connections. Finally, the output image is generated. The limitation is that traditional approaches use fixed filter size, we argue that this could be further improved by a multi-scale approach, as shown in Figure 3.17. The current Joint Exploration Model (JEM) 7.1 software uses minimum coding unit (Min CU) of 2×2 or 4×4. The height and width of each frame must be divided by either 2 or 4. Thus, we use even downsampling factors (2 and 4) instead of odd ones such as 3 or 5.

Our problem can be described by given a low resolution image I^{LR}, reconstruct a high

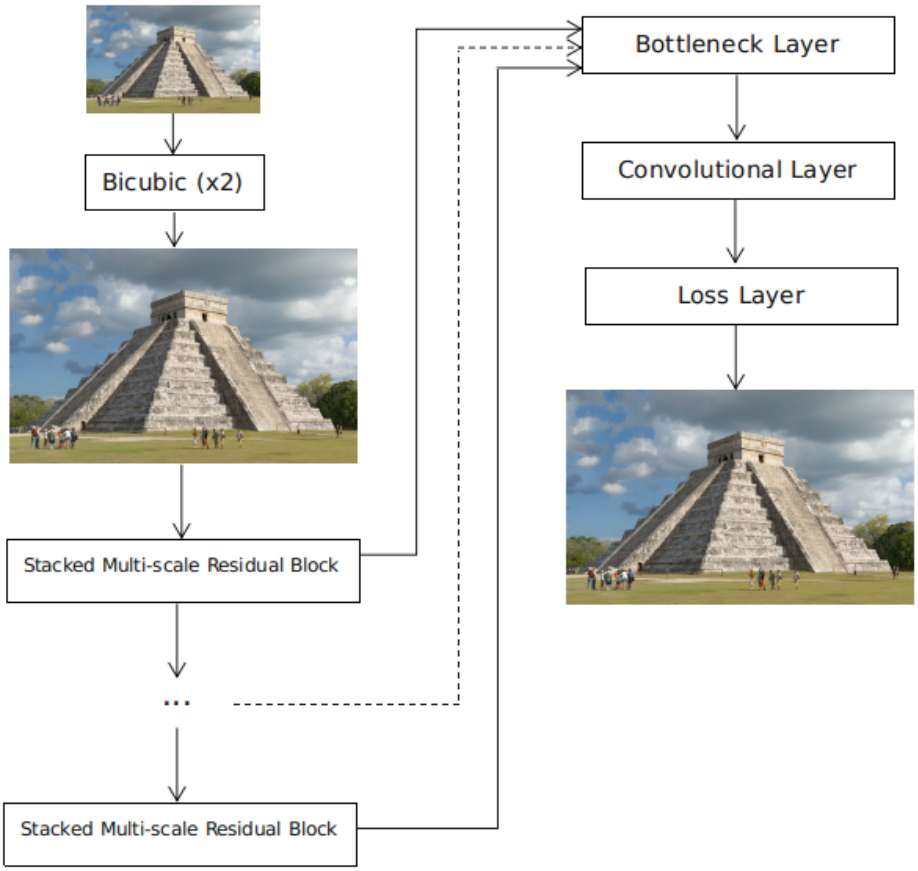


Figure 3.17: Overview of the proposed multi-scale network (MSN) architecture.

resolution one I^{SR}. I^{LR} is the reconstructions obtained by JEM 7.1 with bicubicl downsample of original frame as input. Our goal is to find a non-linear mapping which inputs I^{LR} and outputs I^{SR}. Given a training dataset $\{I_i^{LR}, I_i^{HR}\}$, solve the following minimization problem [55]:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}^{SR}(F_\theta(I_i^{LR}), I_i^{HR}), \tag{3.7}$$

where θ denotes the weights and biases of each neural network layer, I^{HR} is the ground truth high resolution image, and $\mathcal{L}$ is the L2 loss function to minimize the difference between I_i^{SR} and I_i^{HR}, the L2 loss function can be described as follows:

$$\mathcal{L}(F_\theta(I_i^{LR}), I_i^{HR}) = \left\| F_\theta(I_i^{LR}) - I_i^{HR} \right\|_2 . \tag{3.8}$$

Figure 3.17 illustrates the proposed network architecture. Our network takes a 1/2 resolution reconstruction as input. The input is bicubicly interpolated and upsampled by a factor of 2. We choose bicubic pre-upsampling for the following two reasons: First, it speed up the training process without performing the deconvolution operation. Second, we use Caffe platform due to it's C++ implementation is much easier to be incorporated into JEM software, and pixel shuffle is not available in current version of Caffe. Our network takes the bicubicly interpolated patch of 1/2 resolution reconstruction, followed by a sequence of stacked multi-scale residual blocks. These residual blocks with skip connection could avoid the vanishing gradient problem which requires long term memory. Then, each intermediate feature extracted by residual blocks is sent to the bottleneck layer via concatenation, followed by a 1×1 convolution operation. It can be described by the following equation:

$$F_{HR} = \omega * [F_1, F_2, F_3, ..., F_N] + b, \tag{3.9}$$

where F_i represents the output of the i^{th} residual block, and $[F_1, F_2, F_3, ..., F_N]$ denotes the concatenation operation.

The Multi-Scale Residual Blocks

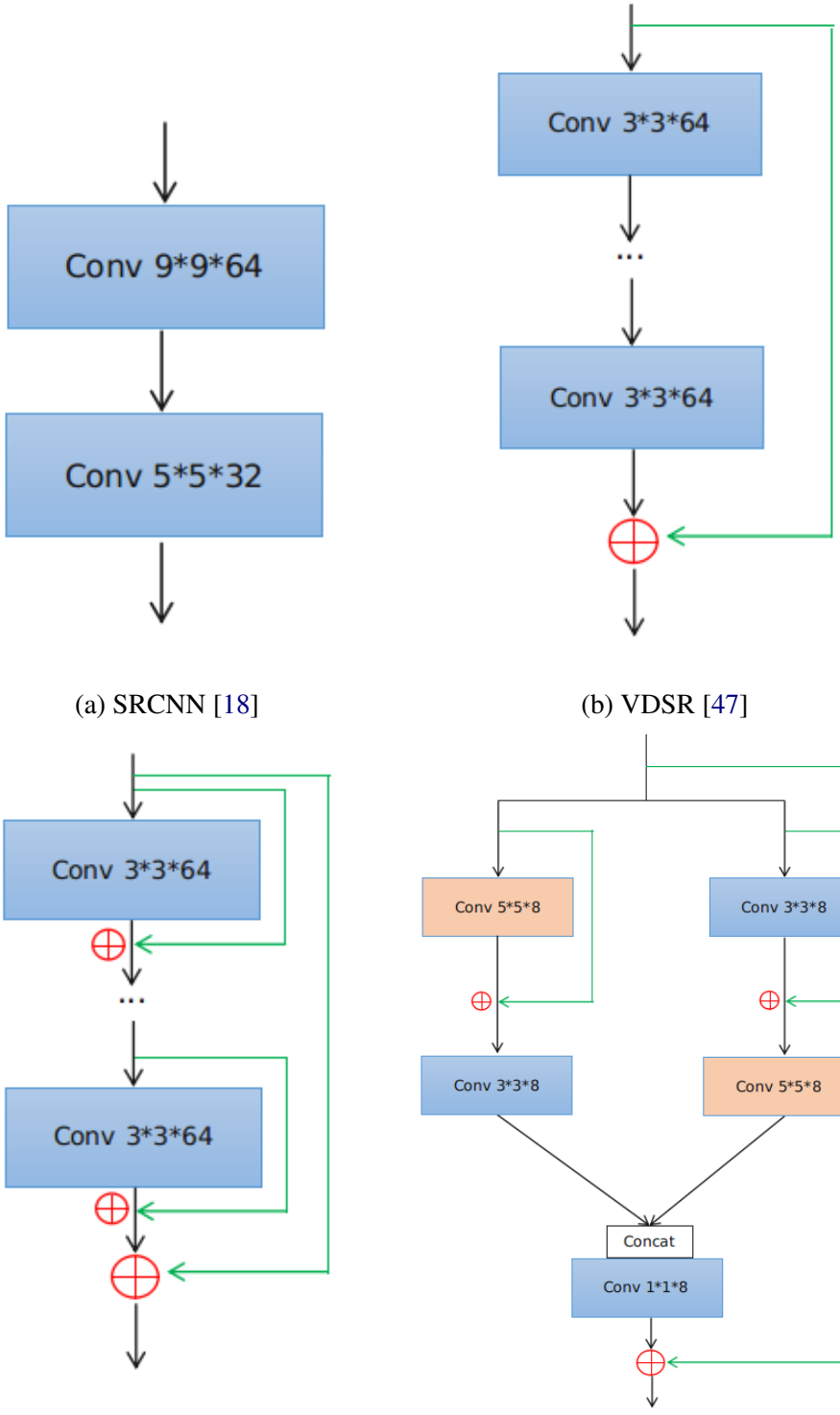
The initial learning-based approaches toward single image SR is based on direct learning of a non-linear mapping without learning residuals such as [18]. Then researches find that as the number of layers increases, the learning became not very effective, the so-called vanishing gradient problem. Thus, residual learning structure is proposed to learn a structural residual such as the one used in [47] with one skip connection. Afterwards, multiple local skip connection is proposed such that each local block features a residual-like structure, as shown in Figure 3.18c.

Different from previous ones, we design the residual block with two channels, each of which has image feature detected at different scales. As shown in Figure 3.18d, for each residual block, the input is followed by two separate convolutional layers: one has kernel size of 5×5, and the other has 3×3. Each convolutional layer learns a residual, added to the input feature map, followed by a convolutional layer at different scale. For example, the first 5×5 is followed by 3×3, vice versa. Then, we concatenate the feature maps and use a 1×1 along with element wise summation. We argue that the proposed residual block is more effective than previous ones in terms of PSNR performance, and we will discuss the implementation detail in the result section.

3.3.3 Experimental Results

The CNN Architecture

Figure 3.19 shows part of our final implementation of the network. $data$ refers to the patches from 1/2 reconstruction. Each data entry is followed by a $1 \times 1 \times 8$ convolutional layer ($conv1\_1, conv3\_1$). For each residual block, we firstly use $3 \times 3 \times 8$ and $5 \times 5 \times 8$ convolutional layer ($conv1\_3\_1, conv1\_5\_2$) followed by rectified linear unit (ReLu) which adds non-linearity to the network. Then, we perform element wise summation between the convolutional entry and residual ($sum1\_1, sum1\_2$), followed by convolutional layers



(a) SRCNN [18]

(b) VDSR [47]

(c) SRGAN [52]

(d) The proposed multi-scale residual block.

Figure 3.18: Comparison of residual blocks among precious approaches and the proposed one. Our residual block has feature maps learned from different scales, followed by concatenation and convolution.

with different size of filters ($conv1\_5\_1, conv1\_3\_2$). The result from previous layer are concatenated together ($concat1$), followed by a $1 \times 1 \times 8$ convolutional layer. There are eight residual blocks in total. Due to space limit, we only show two out of eight residual blocks in Figure 3.19.

The Training and Testing dataset

As shown in Table 3.4, most of the previous approaches use the following dataset for training and testing such as Set5 [9], Set14 [108], Urban100 [36], B100 [95], and MANGA109 [64]. We propose to use the DIV2K [2] dataset with high quality images (originally constructed for image super-resolution). It includes 800 $2k$ resolution images for training, and another 100 images for testing and validation.

Implementation Details

The current JEM 7.0 software takes a YUV sequence along with it's configuration file as input. Thus,

- Step 1: We transform the PNG image file in DIV2k dataset into a YUV sequence with one single frame. We generate the configuration file based on the image size. For 1/2 resolution, we downsample the image with bicubic operation, and transform it into a smaller YUV sequence.
- Step 2: We encode the YUV sequences using JEM 7.1 software, with deblocking filter, bilateral filter, sample adaptive offset, and adaptive loop filter turned on.
- Step 3: Once we obtain the reconstruction as a YUV file, we transform it back to PNG file, and use bicubic operator to pre-upsample it.
- Step 4: The original and recon (label and input) are divided into 40×40 patches, with step size being 60.

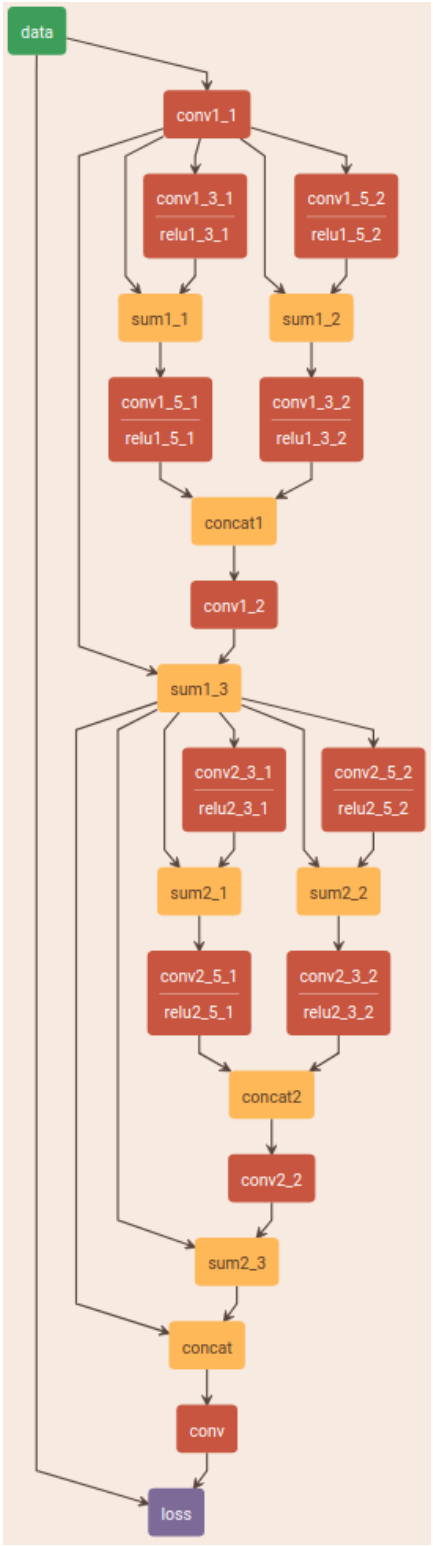


Figure 3.19: Implementation detail of the proposed multi-scale architecture. This figure only shows two stacked multi-scale residual blocks due to space limit, we use eight residual blocks in the experiment.

- Step 5: The training patch is used to train the Caffe model which could be integrated into JEM 7.0 software directly.

We choose Caffe [42] instead of TensorFlow [1] or Pytorch [76] because Caffe's C++ open source implementation is much easier to integrate with JEM software which is also written in C++. We choose $Adam$ solver with base learning rate being 0.0001. The learning rate decays every 500000 steps with gamma ratio 0.1. We train the model with 1 million steps in GPU mode, and it takes around 10 hours to finish on a work station with CPU Intel i7-7920X and GPU NVIDIA Titan Xp.

After obtaining the Caffe model, we integrate it into JEM software. We design the CNN SR operation to be right after the JEM loop filters. Thus, it serves as a post-processing module in both encoder and decoder side. We record the PSNR and bitrate which will be shown in next subsection.

Bitrate and PSNR Performance

Table 3.5 shows the bitrate performance in kilobytes per second for 100 testing sequences encoded under original resolution with QP 51, 1/2 resolution with QP 44, 1/4 resolution with QP 37, and 1/8 resolution with QP 28, respectively. Figure 3.20 shows the bitrate value in bar plot for 30 testing sequences. From Table 3.5 and Figure 3.20, it is obvious that we successfully control the rate by using different QP. Thus, for each sequence encoded under different resolution, the bitrate is very close. Notice that the bitrate of downsampled sequence also includes the bits consumed by the CNN model. In fact, our CNN model is only 56.2 KB thanks to the light weight of the proposed CNN architecture.

We compare MSN against several baseline and other approaches. They are baseline approach, 2x bicubic upsampling, 2x MSN, 4x bicubic upsampling, 4x MSN, SRCNN* [18], VDSR* [47], and SRGAN_G* [52], respectively. Note that we did not use the original SRCNN, VDSR, and SRGAN provided by the authors, as the model is too large to be implemented in our low bandwidth scenario. We replace our residual block by the ones

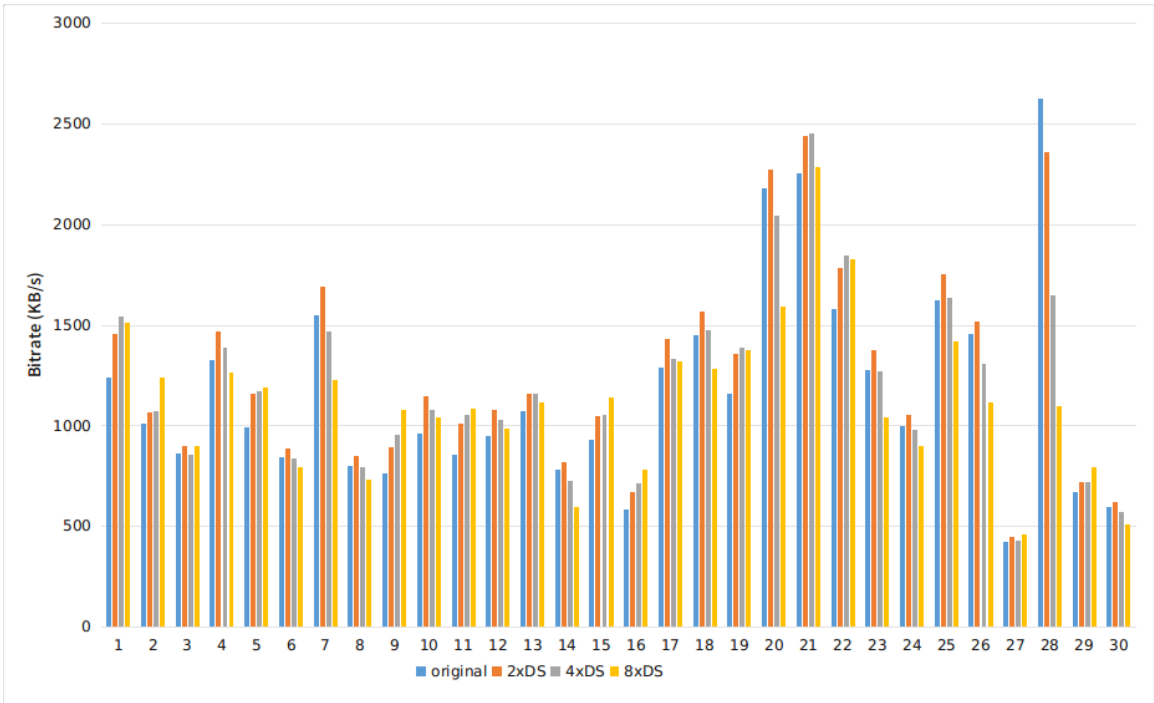


Figure 3.20: Bar plot of bitrate performance in kilobytes per second for 30 testing sequences encoded under original resolution, 1/2 resolution, 1/4 resolution, and 1/8 resolution, respectively. The bitrate of downsampled sequence includes the number of bits consumed by our CNN model.

proposed in SRCNN, VDSR and SRGAN (as shown in Figure 3.18), and control the model depth to ensure each model has a size of around 60 KB. Then, we trained each model on our dataset to make the comparison a fair game. Only the generative module of SRGAN is used in the comparison since PSNR is the key performance measure in video coding.

Table 3.6 compares the PSNR performance of 50 testing sequences. Figure 3.21 shows the average PSNR in bar plot. There are some observations and conclusions we can draw from the results:

1. The proposed 2x MSN network achieves the best performance (26.60 dB) on the testing dataset.

2. Our 2x MSN is better than SRCNN*, VDSR*, and SRGAN_G*. This proves that the proposed multi-scale residual block with concatenation of feature maps is more superior than previous approaches as shown in Figure 3.18.

3. 2x Bicubic, 2x MSN, 4x MSN, SRCNN*, VDSR*, and SRGAN_G* are better than baseline approach.

4. 8x MSN and 8x Bicubic are worse than baseline approach. This confirms that 8x downsampling has too much information loss.

Visual Quality Comparison

Figure 3.22 shows the visual quality comparison among original image, baseline results, and ours. The images feature a variety of styles including portraits, fine-detailed images, architecture, and scenery. It can be seen from Figure 3.22 that our results has less distortion, less blurriness, and less blocking artifacts than baseline results thanks to the rate control and the efficient network architecture.

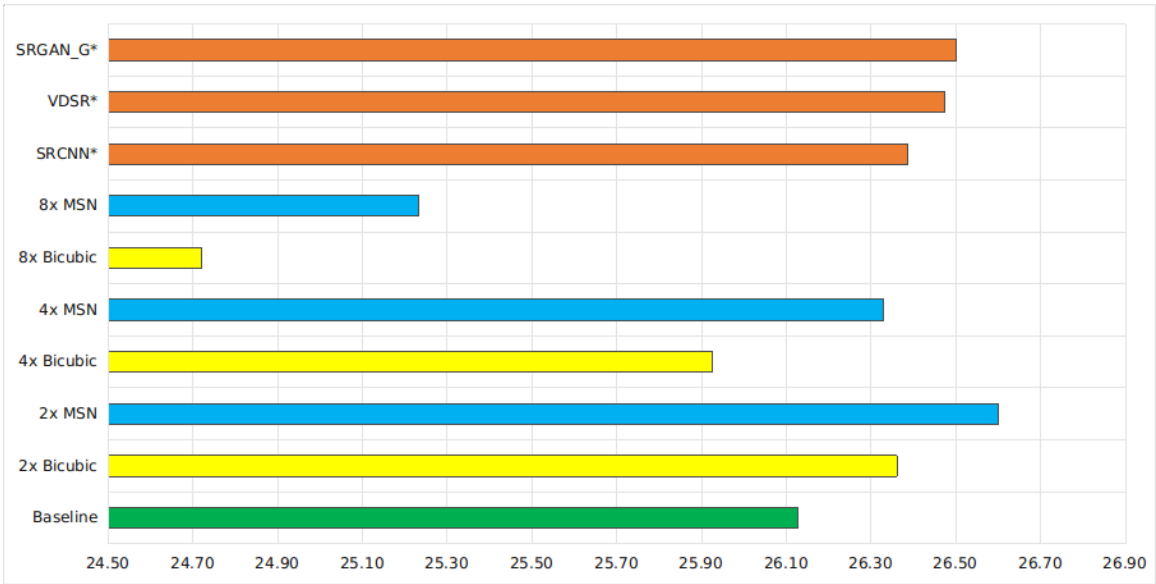


Figure 3.21: Average PSNR performance in dB for baseline approach, 2x bicubic upsampling, 2x MSN, 4x bicubic upsampling, 4x MSN, SRCNN [18], VDSR [47], and SRGAN [52], respectively.

3.3.4 Conclusion and Future Work

In this section, we propose a multi-scale super-resolution framework to improve the image quality under low bandwidth in HEVC. The current HEVC software uses very large QP to encode the sequence resulting in heavy artifacts in reconstruction such as blurriness and blocking artifacts. We down-sample the original sequence by a factor of two, the down-sampled sequences are then used as input for JEM software to encode with a smaller QP (smaller compression loss), finally, the original resolution reconstruction is obtained by up-sampling the 1/2 resolution recons with the proposed MSN network. We show numerically that our framework wins the bitrate-distortion trade-off. Also from a visual quality point of view, our results has less distortion and artifacts thanks to the efficient MSN network.

Regarding the future work, traditional SR networks takes one bicubic downsampled image as input, followed by either bicubic pre-upsamling, or deconvolution, or pixel shuffle to bring the input image back to the original size in a direct or progressive manner. Direct approach enlarges the image to the desired size in one step, while progressive approach

(a) Original (b) Baseline (c) Ours

Figure 3.22: Visual comparison among original frame, baseline result and our result.

use intermediate upsampling factors to achieve the desired resolution. Then, some residual learning steps is fulfilled by convolutional layers and rectified linear unit (ReLu) with skip connections. Finally, the output image is generated. We argue that this one-to-one non-linear mapping might be further improved by a multiple-to-one approach, as shown in Figure 3.23. Instead of taking only one image/patch as input, the network could take two input images/patches with different scales at a time. Rate control with two inputs is an interesting topic to explore in the future.

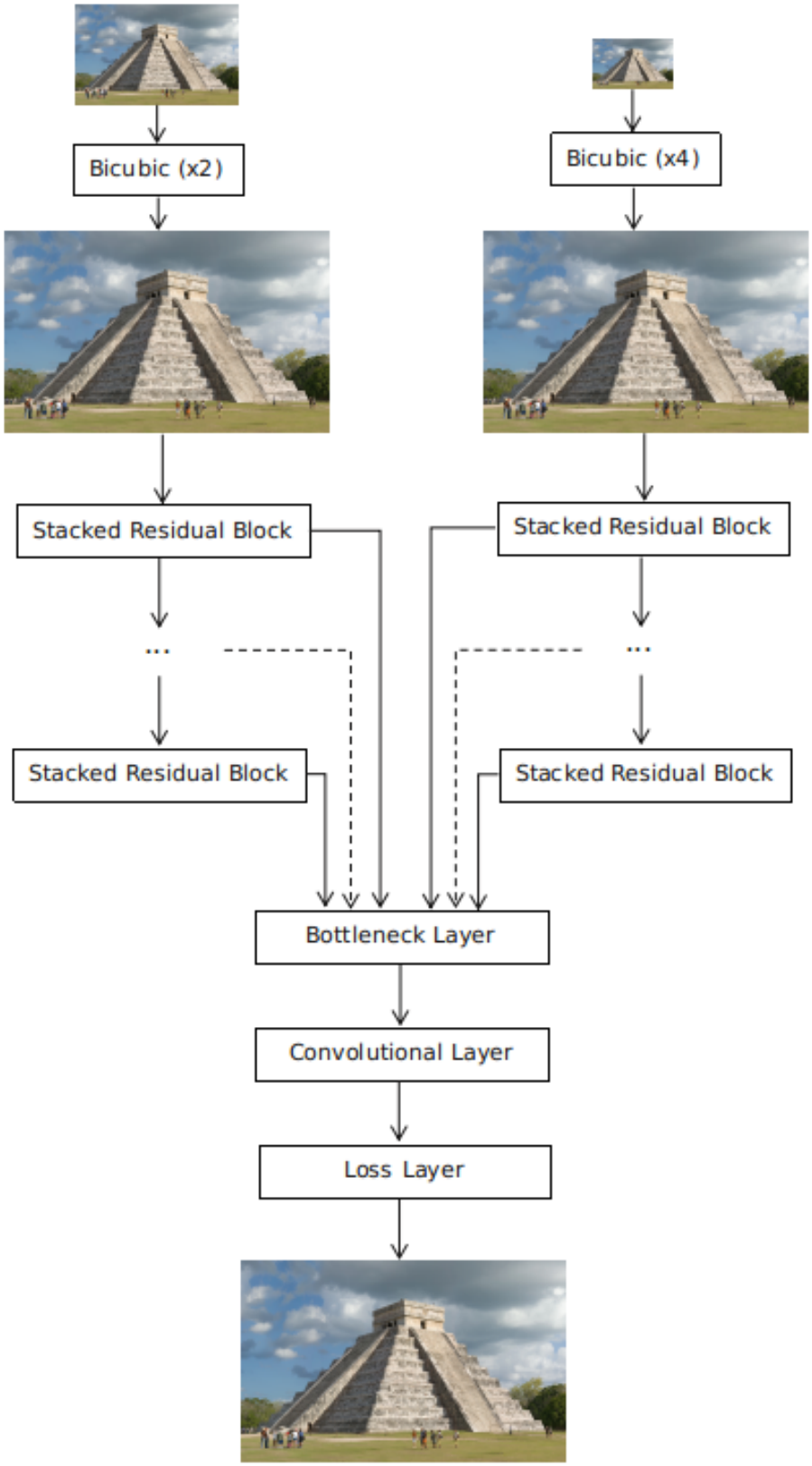


Figure 3.23: Overview of the potential super-resolution architecture which takes two input at a time.

Table 3.3: BD-rate gain computed using bit rate and PSNR.

		baseline				ours				BD-rate (piecewise cubic)		
	QPISlice	kbps	Y psnr	U psnr	V psnr	kbps	Y psnr	U psnr	V psnr	Y	U	V
BQMall_22	22	27456.96	41.87	43.82	45.45	26777.76	41.88	43.73	45.34	-9.31%	-6.02%	-6.09%
BQMall_27	27	17188.32	38.84	41.69	42.99	16638.72	39.27	41.79	43.12			
BQMall_32	32	10369.44	35.72	40.02	41.04	10037.76	36.20	40.19	41.25			
BQMall_37	37	5836.32	32.44	38.81	39.74	5706.24	32.85	38.88	39.76			
BQSquare_22	22	12614.88	41.80	44.09	44.92	12241.44	41.79	43.96	44.98	-8.54%	-3.72%	-4.91%
BQSquare_27	27	8368.32	37.37	41.52	42.58	8144.64	38.03	41.51	42.59			
BQSquare_32	32	5291.04	33.83	39.86	40.72	5180.64	34.42	40.04	40.96			
BQSquare_37	37	3306.24	30.50	39.15	39.70	3248.64	30.96	39.14	39.73			
BQTerrace_22	22	170128.32	41.92	43.01	44.71	169805.28	42.25	43.01	44.71	-6.09%	-3.24%	-3.43%
BQTerrace_27	27	90831.36	37.82	41.05	43.01	89783.52	38.08	41.14	43.05			
BQTerrace_32	32	49946.88	34.93	39.62	41.73	48916.80	35.19	39.64	41.78			
BQTerrace_37	37	27516.96	32.19	38.51	40.78	26996.16	32.41	38.53	40.84			
BasketballDrill_22	22	17948.00	42.10	44.20	45.25	17223.60	42.19	44.48	45.62	-9.04%	-12.68%	-16.13%
BasketballDrill_27	27	9466.00	38.96	42.10	42.81	9229.60	39.32	42.43	43.38			
BasketballDrill_32	32	5048.40	36.18	40.36	41.11	4857.60	36.46	40.64	41.48			
BasketballDrill_37	37	2843.60	33.70	39.16	39.63	2736.80	33.88	39.28	39.82			
BasketballDrive_22	22	41873.60	41.12	45.44	47.24	42980.80	41.17	45.49	47.28	-5.29%	-8.47%	-11.60%
BasketballDrive_27	27	18688.00	39.59	44.39	45.51	18480.80	39.65	44.51	45.80			
BasketballDrive_32	32	10092.40	38.27	43.27	43.92	9852.00	38.40	43.40	44.20			
BasketballDrive_37	37	5750.40	36.64	42.37	42.62	5648.80	36.81	42.56	42.96			
BasketballPass_22	22	5122.00	42.93	45.23	45.56	4983.60	42.93	45.30	45.55	-9.28%	-5.06%	-8.20%
BasketballPass_27	27	3119.60	39.55	42.67	42.64	2980.80	39.87	42.67	43.03			
BasketballPass_32	32	1786.00	36.22	40.62	40.64	1710.80	36.67	40.65	40.64			
BasketballPass_37	37	1014.80	33.25	38.84	38.70	963.20	33.47	39.00	38.90			
BlowingBubbles_22	22	9193.20	41.21	42.23	44.15	8979.20	41.55	42.41	44.35	-9.10%	-8.84%	-8.13%
BlowingBubbles_27	27	5494.80	37.38	39.36	41.24	5332.00	37.90	39.63	41.56			
BlowingBubbles_32	32	3074.40	34.02	37.09	39.12	2984.00	34.37	37.35	39.23			
BlowingBubbles_37	37	1623.20	30.76	35.46	37.61	1604.40	31.10	35.70	37.83			
Cactus_22	22	103709.60	40.72	41.39	44.04	106302.40	40.89	41.34	44.12	-5.19%	-3.86%	-7.42%
Cactus_27	27	47574.00	38.10	39.52	42.21	46602.80	38.19	39.56	42.35			
Cactus_32	32	26058.80	36.04	38.59	40.64	25436.80	36.21	38.70	40.80			
Cactus_37	37	14284.80	33.74	37.87	39.49	13962.40	33.88	37.96	39.65			
FourPeople_22	22	27720.00	44.23	47.20	48.45	26996.16	43.84	47.22	48.48	-6.48%	-5.74%	-6.66%
FourPeople_27	27	17287.20	41.89	45.28	46.56	16730.88	42.04	45.42	46.71			
FourPeople_32	32	10730.40	39.20	43.62	44.92	10391.04	39.54	43.68	45.04			
FourPeople_37	37	6635.52	36.21	42.42	43.67	6424.80	36.51	42.52	43.81			
Johnny_22	22	17940.96	44.32	48.41	48.98	17580.48	44.10	48.50	49.04	-4.85%	-9.49%	-7.67%
Johnny_27	27	9917.76	42.36	46.85	47.56	9610.08	42.41	47.07	47.68			
Johnny_32	32	5746.56	40.30	45.40	46.00	5569.92	40.48	45.57	46.16			
Johnny_37	37	3398.88	37.97	43.99	44.87	3303.36	38.09	44.19	44.94			
Kimono_22	22	15739.58	43.22	44.44	46.15	15648.19	43.10	44.41	46.19	-1.00%	-0.79%	-2.68%
Kimono_27	27	9147.84	42.06	42.71	43.91	9101.38	42.06	42.72	43.99			
Kimono_32	32	5624.83	40.34	41.35	42.36	5585.86	40.40	41.37	42.43			
Kimono_37	37	3454.08	38.00	40.36	41.30	3433.73	38.02	40.37	41.36			
KristenAndSara_22	22	20059.20	44.80	47.87	48.68	19622.88	44.32	47.90	48.84	-3.78%	-7.26%	-7.13%
KristenAndSara_27	27	12044.16	42.68	46.01	47.04	11748.48	42.68	46.20	47.29			
KristenAndSara_32	32	7338.24	40.27	44.39	45.67	7186.08	40.50	44.58	45.72			
KristenAndSara_37	37	4537.92	37.59	43.27	44.48	4443.36	37.84	43.40	44.55			
ParkScene_22	22	46036.99	41.82	43.19	44.03	45539.90	41.93	43.22	44.11	-6.22%	-1.66%	-2.03%
ParkScene_27	27	25056.19	39.12	41.09	41.56	24545.09	39.34	41.06	41.55			
ParkScene_32	32	13223.23	36.36	39.47	40.23	12963.07	36.57	39.50	40.25			
ParkScene_37	37	6591.36	33.59	38.35	39.57	6490.37	33.73	38.32	39.59			
PartyScene_22	22	49524.00	41.14	42.09	42.81	48264.00	41.44	42.16	42.92	-9.56%	-5.88%	-5.83%
PartyScene_27	27	31902.80	36.75	39.14	39.77	30690.80	37.35	39.25	39.92			
PartyScene_32	32	19292.40	33.00	37.11	37.61	18734.80	33.55	37.22	37.76			
PartyScene_37	37	10673.60	29.38	35.58	36.11	10396.80	29.74	35.79	36.08			
PeopleOnStreet_22	22	95945.28	43.74	46.08	45.46	94836.24	42.77	46.14	45.51	1.96%	-5.08%	-4.82%
PeopleOnStreet_27	27	55698.72	40.53	44.13	44.02	54637.20	40.18	44.24	44.10			
PeopleOnStreet_32	32	31605.84	37.48	42.54	42.76	30979.68	37.50	42.65	42.83			
PeopleOnStreet_37	37	18281.04	34.69	41.32	41.89	17975.76	34.82	41.40	41.91			
RaceHorsesC_22	22	16684.56	42.41	43.16	43.82	16380.00	42.18	43.22	43.92	-4.94%	-3.92%	-8.20%
RaceHorsesC_27	27	10274.64	38.79	40.18	41.34	10005.12	38.97	40.22	41.53			
RaceHorsesC_32	32	5988.24	35.29	38.07	39.71	5863.44	35.56	38.16	39.94			
RaceHorsesC_37	37	3130.56	31.78	36.78	38.25	3086.40	31.98	36.87	38.49			
RaceHorses_22	22	4984.32	42.74	43.08	43.76	4870.32	42.71	43.31	44.09	-6.48%	-8.16%	-8.66%
RaceHorses_27	27	3154.08	38.68	40.12	41.04	3075.12	39.05	40.44	41.34			
RaceHorses_32	32	1841.76	34.83	38.03	38.49	1813.20	35.22	38.25	38.92			
RaceHorses_37	37	1034.88	31.48	36.43	36.81	988.80	31.64	36.63	36.74			
Traffic_22	22	93635.52	43.77	42.78	44.93	92242.80	43.45	42.68	44.99	-3.82%	-2.93%	-4.96%
Traffic_27	27	52684.80	40.76	40.46	42.60	51528.48	40.82	40.51	42.70			
Traffic_32	32	30082.80	37.82	38.88	40.95	29382.48	38.03	38.91	41.04			
Traffic_37	37	16979.76	34.84	37.81	39.84	16688.40	35.03	37.86	39.94			
All										-5.94%	-5.71%	-6.92%

Table 3.4: Comparison of previous works on single image super-resolution.

Approach	Year	Layers	Model Size	Residual Learning	Reconst-ruction	Loss func-tion	Training	Testing
SRCNN [18]	2014	3	32.4KB	No	Direct	L2	91 Images	Set5, Set14
SCN [99]	2015	5	628KB	No	Progressive	L2	91 Images	Set5, Set14, BSD100 [63]
FSRCNN [20]	2016	8	51.3KB	No	Direct	L2	91 images, General-100	Set5 [9], Set14 [108], BSD200 [63], BSD500 [63]
VDSR [47]	2016	20	2.37MB	Yes	Direct	L2	91 images, 291 images [86]	Set5 [9], Set14 [108], Ur-ban100 [36], B100 [95]
DRCN [48]	2016	29	2.87MB	No	Direct	L2	91 images	Set5 [9], Set14 [108], Ur-ban100 [36], B100 [95]
LapSRN[51]	2017	27	2.37MB	Yes	Progressive	Charbonnier	91 images, 200 images [3]	Set5 [9], Set14 [108], Ur-ban100 [36], B100 [95], MANGA109 [64]
EDSR [58]	2017	32	3.5MB	Yes	Direct	L1	DIV2K [22]	DIV2k [22], Set5 [9], Set14 [108], Urban100 [36], B100 [95]
SRGAN [52]	2017	33	8.7MB	Yes	Direct	Adversarial	DIV2k [22]	Set5 [9], Set14 [108], B100 [95], BSD300 [63]
MSRB [55]	2018	45	22.5MB	Yes	Direct	L1	DIV2k [22]	Set5 [9], Set14 [108], Ur-ban100 [36], B100 [95], MANGA109 [64]
RCAN [112]	2018	400	59.88MB	Yes	Direct	L1	DIV2k [22]	Set5 [9], Set14 [108], Ur-ban100 [36], B100 [95], MANGA109 [64]
MSCN [35]	2018	100	N/A	Yes	Direct	L2	91 images, BSD300 [63]	Set5 [9], Set14 [108], Ur-ban100 [36], B100 [95]

Table 3.5: Bitrate performance in kilobytes per second for 100 testing sequences encoded under original resolution, 1/2 resolution, 1/4 resolution, and 1/8 resolution, respectively. Notice that the bitrate of downsampled sequence includes the number of bits consumed by our CNN model.

Original	2xDS	4xDS	8xDS
1237.12	1454.56	1537.92	1508.64
1007.84	1065.6	1072	1236.8
862.56	899.36	854.88	900
1325.76	1469.44	1386.88	1264
989.44	1160	1166.4	1190.08
840.64	883.36	832.96	794.4
1548.32	1687.2	1469.44	1225.28
795.52	846.4	791.84	730.88
762.56	888	951.36	1073.92
959.04	1141.92	1074.56	1039.36
853.92	1005.92	1052.64	1084.96
948.64	1074.56	1026.24	982.88
1067.68	1157.92	1154.24	1111.68
778.56	817.92	726.4	592.8
930.24	1045.92	1048.96	1137.76
579.52	668.64	710.08	781.44
1287.04	1432.16	1331.04	1319.36
1445.6	1563.84	1469.76	1282.88
1156.16	1352.96	1383.36	1376.16
2180.8	2273.28	2044.16	1591.52
2254.72	2435.52	2448.64	2284.8
1578.4	1783.52	1845.76	1828.16
1272.48	1372.8	1265.92	1040.48
993.28	1050.56	975.36	898.72
1620	1753.28	1632.8	1418.72
1451.36	1518.24	1307.04	1112.16
419.52	445.12	427.04	456.64
2625.6	2356.48	1645.44	1092.64
670.88	717.28	719.68	788.96
595.68	617.92	572.32	509.92
1246.08	1379.36	1330.4	1289.6
1333.6	1485.28	1484.96	1361.12
671.52	724.16	679.36	655.52
901.6	1000.16	952.48	909.92
1509.6	1717.28	1635.52	1493.12
1615.68	1624	1377.6	1198.56
1856.8	2031.84	1933.44	1669.92
356	359.84	374.4	470.88
612.8	686.56	716.32	769.28
1101.76	1295.04	1377.92	1433.76
1033.76	1120.16	1069.28	975.04
736.64	814.08	838.72	880.16
237.6	233.92	232	260.16
320.8	328.64	333.76	393.12
1374.08	1328.8	1090.24	846.56
1577.76	1504	1255.68	1034.08
1397.6	1516.8	1396	1277.6
1369.12	1461.28	1353.28	1336.16
3620.16	3799.2	3281.28	2318.24
948.8	1020.64	993.76	976.16

Original	2xDS	4xDS	8xDS
1027.2	1072	998.24	844.8
326.24	373.28	389.6	470.72
715.84	783.36	714.56	684.96
1144	1248.16	1189.76	1007.04
998.08	1101.76	1113.28	1065.6
997.6	1183.2	1020.32	990.56
386.88	398.08	399.04	455.68
907.68	1045.12	1086.88	1153.92
1308.32	1500	1492	1347.84
2754.72	3067.52	2712.16	2130.72
1790.88	1845.92	1562.08	1235.84
272	292.48	318.4	392.32
718.08	767.52	778.72	848.32
651.04	752.8	799.84	907.36
865.76	924.48	932.8	1011.36
1237.76	1449.76	1455.04	1434.56
1229.44	1368	1297.44	1209.44
515.36	537.28	497.12	429.12
1231.36	1286.72	1113.28	995.04
1965.12	2264.16	2211.04	2056.16
1086.08	1205.44	1206.24	1235.68
1828.16	1989.12	1858.24	1639.2
1595.52	1688	1574.08	1322.72
616.16	702.24	719.04	790.72
2175.52	2331.52	2145.12	1873.92
1241.6	1396.32	1303.04	1211.36
474.08	511.84	525.6	620.64
1313.6	1467.04	1519.52	1525.28
1333.6	1395.52	1270.56	1075.68
601.28	679.84	705.28	724.96
709.12	741.92	689.76	665.12
1000.48	1066.56	1049.44	1117.6
1316.8	1518.4	1514.08	1395.36
1303.04	1449.12	1380.32	1332.16
1058.24	1126.56	981.44	874.88
624.16	660.8	632.8	646.24
1207.52	1330.72	1223.84	1052
1397.6	1505.44	1501.44	1479.36
2028.16	2289.6	2198.72	1977.76
706.24	765.92	729.76	731.68
1799.36	1919.84	1773.76	1627.36
1660.96	1756.8	1682.88	1398.72
749.28	804.16	823.04	888.48
642.72	705.28	713.92	762.4
1984	2222.24	1900.64	1433.12
447.2	462.4	464.96	475.2
2472	2770.72	2592.16	2067.68
699.04	773.92	733.76	649.76
1129.92	1230.56	1201.44	1235.2
1347.2	1410.08	1255.2	1077.6

Table 3.6: PSNR performance in dB for 50 testing sequences, from left most to right most: baseline approach, 2x bicubic upsampling, 2x MSN, 4x bicubic upsampling, 4x MSN, SRCNN [18], VDSR [47], and SRGAN [52], respectively.

Baseline	2x Bicubic	2x MSN	4x Bicubic	4x MSN	8x Bicubic	8x MSN	SRCNN*	VDSR*	SRGAN_G*
25.00	25.50	25.58	25.50	25.64	24.53	24.86	25.60	25.75	25.58
29.59	30.16	30.35	30.41	30.67	29.81	30.24	30.15	30.23	30.30
31.56	32.24	32.77	31.77	32.83	29.79	31.17	32.22	32.44	32.24
24.90	25.20	25.40	24.80	25.20	23.48	24.08	25.20	25.42	25.32
25.61	26.02	26.03	25.94	25.99	25.26	25.36	26.12	26.14	26.14
26.73	26.89	27.10	26.39	26.70	25.50	25.77	26.96	27.01	26.99
20.05	20.06	20.18	19.75	19.90	19.25	19.34	20.12	20.20	20.09
25.85	25.94	26.11	25.50	25.86	24.50	24.89	25.92	25.96	26.05
26.93	27.35	27.39	27.54	27.61	27.20	27.26	27.31	27.41	27.35
24.78	25.17	25.19	24.97	25.03	24.44	24.51	25.25	25.29	25.21
25.61	26.03	26.09	26.03	26.13	25.38	25.53	26.08	26.10	26.25
26.01	26.16	26.45	25.26	25.80	23.78	24.34	26.18	26.23	26.22
27.72	28.09	28.36	27.74	28.25	26.15	26.96	28.14	28.09	28.18
27.21	27.35	27.57	26.62	26.96	25.26	25.57	27.30	27.36	27.51
28.73	29.38	29.51	29.44	29.70	28.33	28.73	29.46	29.46	29.51
27.60	28.04	28.17	27.99	28.23	27.33	27.57	28.12	28.08	28.09
27.59	28.16	28.29	28.13	28.34	26.99	27.43	28.15	28.16	28.44
26.03	26.29	26.54	25.71	26.16	24.10	24.63	26.26	26.47	26.49
24.73	25.14	25.21	25.11	25.25	24.36	24.62	25.20	25.30	25.32
23.36	23.26	23.71	21.86	22.87	19.52	20.57	23.29	23.36	23.55
26.00	26.22	26.96	24.59	26.26	20.36	22.15	26.20	26.27	26.36
25.54	26.09	26.23	25.97	26.21	24.59	25.10	26.04	26.12	26.16
24.66	24.79	24.97	24.31	24.59	23.13	23.43	24.77	24.95	24.83
26.48	26.48	26.98	25.51	26.36	23.81	24.65	26.47	26.67	26.59
24.92	25.10	25.45	24.35	24.97	22.59	23.35	25.19	25.18	25.18
23.58	23.56	23.74	23.07	23.37	22.12	22.49	23.56	23.60	23.80
30.36	30.59	30.93	29.85	30.33	28.82	29.12	30.60	30.65	30.72
23.45	19.94	22.59	17.10	18.75	15.65	16.20	20.01	20.16	19.96
25.84	26.01	26.07	25.98	26.12	25.34	25.82	26.02	26.12	26.28
24.40	24.43	24.61	23.95	24.19	23.06	23.25	24.45	24.56	24.69
26.36	26.69	27.08	25.95	26.60	24.31	25.16	26.69	26.77	26.94
26.40	26.86	27.08	26.46	26.95	24.63	25.45	26.96	27.08	27.07
30.49	30.99	31.53	29.70	30.81	26.83	27.92	31.05	31.18	31.11
24.48	24.69	24.79	24.42	24.63	23.68	24.03	24.76	24.70	24.75
21.98	22.16	22.21	21.97	22.06	21.44	21.53	22.12	22.22	22.39
25.10	24.78	25.25	23.50	24.19	21.92	22.44	24.87	24.84	24.92
23.73	23.92	24.20	23.42	23.91	21.99	22.56	23.95	23.93	23.95
32.71	33.15	33.21	33.59	33.66	33.81	33.55	33.21	33.13	33.19
26.66	26.97	27.02	26.98	27.09	26.41	26.66	26.99	26.99	27.17
24.94	25.40	25.51	25.49	25.62	24.83	25.09	25.43	25.39	25.55
25.93	26.17	26.35	25.69	25.96	24.54	24.87	26.14	26.23	26.30
26.87	27.22	27.31	27.09	27.27	26.35	26.62	27.17	27.44	27.32
36.04	36.49	36.92	36.18	37.07	34.56	36.17	36.46	36.65	36.60
34.10	34.83	35.25	34.76	35.56	33.47	35.19	34.81	34.92	35.09
23.78	23.30	23.90	22.00	22.67	20.52	20.97	23.26	23.35	23.58
25.36	24.52	25.35	22.64	23.49	20.72	21.16	24.56	24.56	24.71
25.35	25.58	25.86	24.85	25.35	23.23	23.99	25.65	25.76	25.82
25.41	25.51	25.80	24.97	25.45	23.84	24.66	25.60	25.63	25.61
20.91	20.88	21.22	19.98	20.56	18.37	18.95	20.96	21.02	20.88
26.26	26.50	26.75	26.21	26.60	25.19	25.67	26.53	26.49	26.52

Chapter 4

Learning-Based Image Classification for Precision Pollination Robotic Vision

4.1 Introduction

Due to the rapidly increasing of human population and the rigorous demand of high quality food, it is essential to figure out new ways to aid agriculture. Productivity has to be increased to meet the call, while human labors are becoming more and more expensive. Most of agricultural tasks involve repetitive and lengthy which makes robotics a great fit to address the aforementioned problem. Thus, agricultural robotics are rapidly gaining interest in many existing fields such as weed control [83, 90, 62, 61, 67], harvesting [53, 43, 73], yield estimation [38, 71, 69, 77], and quality assessment [101, 82]. These agricultural robotics requires precise vision and autonomy, but only few previous works has addressed these aspects.

The decline of natural pollinators is one of the critical issues faced by the agricultural sector today [72]. Specifically, the decrease of the amount of bees substantially threatens the agricultural production. As a results, local bees is not sufficient for pollination purpose, and farmers have to rent and ship bees from other locations in the United States. It is not

Figure 4.1: The robot "BrambleBee" designed by a group of researchers at WVU for autonomous precision pollination toward bramble plants.

only costly but also unreliable. To resolve this issue, robotic precision pollination technique has been proposed by a group of researchers at West Virginia University (WVU). A fully autonomous precision pollination robot is designed for performing pollination toward bramble plants (*i.e.*,blackberries and raspberries) in a greenhouse. The robot is named "BrambleBee" as shown in Figure 4.1. I was responsible for the computer vision subsystem of the robot, and this chapter will mainly discuss my contribution in robotic vision of "BrambleBee".

In most of the agricultural robotics applications, correctly identifying the plant parts is an important initial step to support autonomous decision making. Specifically in the project of "BrambleBee", the goal is to enable the end-effector installed on the robotic arm to pollinate the blackberry or raspberry flower. Thus, identify the flower and estimating it's pose are two essential steps. Essentially, these two tasks can be effectively addressed using image classification. Thus, we propose to use transfer learning technique to deal with the problems. Transfer learning is a technique in deep learning which reuses the body of a pre-trained network, and simply retrain the last layer using the new data. We collect flower data from a farm and conduct extensive experiments to verify the effectiveness of the proposed technique. The following sections will give an overview of the robotic system,

introduce the transfer learning technique, describe the model selection process, and report experiments and results such as data collection, training/testing details, hyper-parameter tuning and the performance.

4.2 Overview of the Pollination Robot System

We will give a brief overview of the BrambleBee robotic system in this section to help understand how the robot pollinate flower. As shown in Figure 4.1, BrambleBee is a parked vehicle with four wheels built upon a ClearPath Robotics Husky platform. It is equipped with a KONIVA JACO 2 robotic arm mounted in the front side of the vehicle. On the arm, there are two equipment installed: a self designed end-effector for pollinating flowers and an Intel RealSense D435 depth-camera which is used for mapping the local workspace. The robot is powered by batteries.

The robot is designed to work in a greenhouse environment in which plants are arranged in rows. BrambleBee will visit each row, and pollinate the flower on each side. Initially, the robot will construct a map of the greenhouse by traveling across it. Then, it will visit each plants and flowers and perform the pollination procedure. We will mainly discuss the scenario that when the robot is parked in front of a plant and pollinate a flower, because this scenario contains the image classification work I contributed to the project.

When BrambleBee is parked still and in front of a plant, it will start looking for flowers and obstacles via mapping. Searching for flowers is to figure out the pollination sequence, and looking for obstacles is accomplished by depth cameras for the purpose of avoiding damage to the plant and robotic arm. The end-effector will maneuver through a couple of poses, at each pose the end-effector will search for flowers and estimate the corresponding flower pose. After the pose of end-effector is refined using a factor graph based framework, it aligns itself to the center of the flower until contact is made. Then, the precision pollination is executed by releasing pollen to the anthers of the flower. This process is repeated

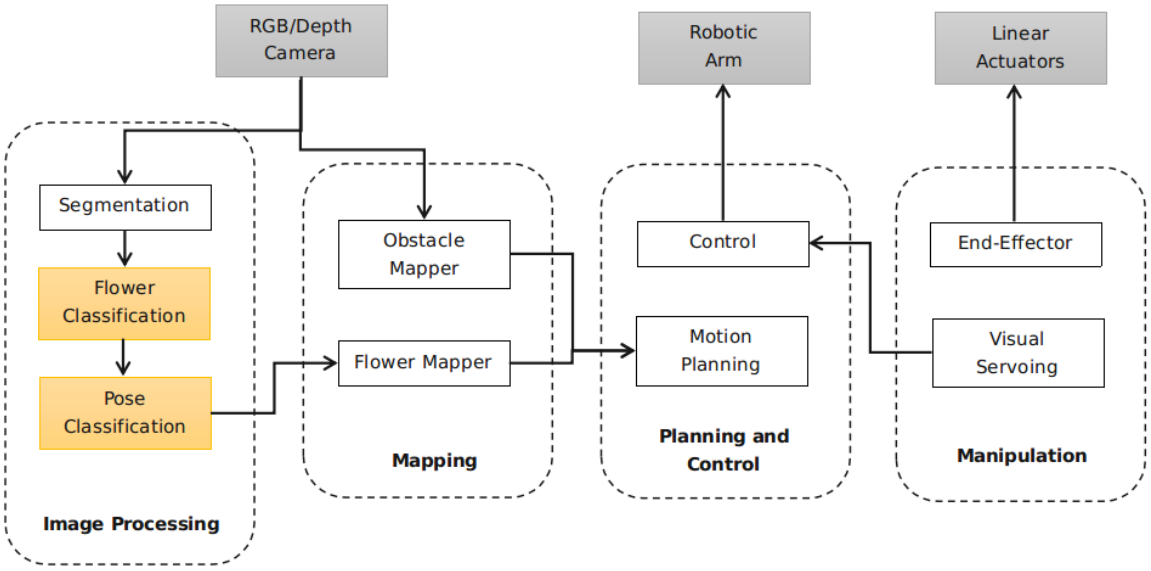


Figure 4.2: An overview of the "BrambleBee" pollination robot software system with four modules: image processing, mapping, planning/control, and manipulation.

until all flowers are correctly pollinated.

As shown in Figure 4.2, the pollination robot software system includes four modules: the image processing module, mapping module, planning and control module, and manipulation module, respectively. The image processing module is responsible for segmenting the flower, refining the segmentation, and estimating the pose of the flower. Mapping module is to map flowers and avoid obstacles. The planning and control module send commands to operate the robotic arm. The manipulation module controls the end-effector to reach and pollinate the flower.

In this project, I have contributed to the flower classification and pose classification part marked in yellow in Figure 4.2, the rest of this Chapter will discuss these two parts in detail, including their motivation, method, implementation details and experimental results.

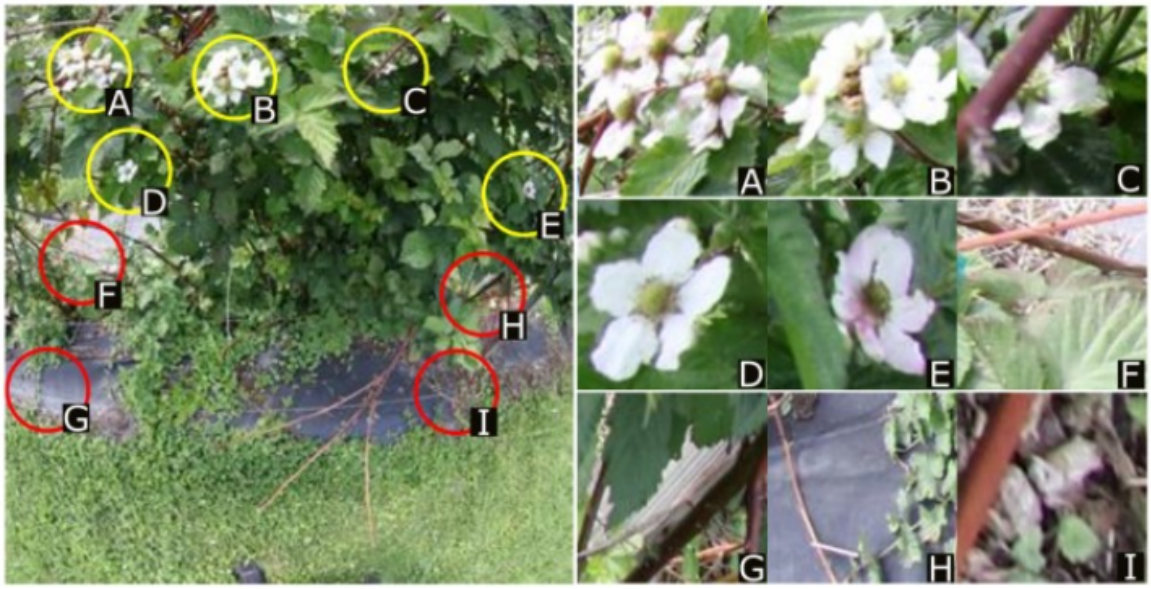


Figure 4.3: Example of the parts of the image extracted using the segmentation algorithm, where A-E are flowers and F-I are non-flowers.

4.3 Image Classification based on Transfer Learning

As shown in Figure 4.2, the image processing module is responsible for segmenting the flower, refining the segmentation, and estimating the pose of the flower. In order to accelerate the segmentation process to make it possible for real-time segmentation, a color look-up table is pre-built under HSV color space. The look-up table consists of all the potential HSV values of a flower. The values are identified empirically from flower images taken in a farm. The look-up table includes colors such as white, white pink, white red, etc. During the segmentation, each image is loaded using OpenCV, and each pixel's HSV value is computed and compared against the look-up table. Once a pixel's color matches the color in the look-up table, we output the pixel resulting in a binary mask. Followed by eroding and dilating, we obtain the initial masks of flowers. Figure 4.3 shows the initial segmentation result. The initial segmentation not only outputs correctly segmented flower patches (A-E), but also a couple of false positives such as patches F-I in Figure 4.3.

Flower and Pose Classification

Segmenting flowers correctly is non-trial since the approach described above tends to produce some false positives when the color falls within the flower color range, such as the cloud's pixels. Moreover, there are some misshapen flowers and buds that do not need pollination. Thus, we need to design a machine learning algorithm to eliminate all those non-flower and incomplete flower pixels. Essentially, this can be fulfilled by a binary classification which tells whether a flower patch contains flower (real) or some non-flower pixels (fake). Transfer learning with Convolutional Neural Networks (CNNs) comes into our mind since it fully reuses the body of a pre-trained classification network, and simply retrain the last layer using the new data. It could not only avoid the lengthy training-from-scratch process, it could also gives good classification performance thanks to the rich features such as edge and shape feature from a pre-trained classification network.

After the flower patches are coarsely segmented with look-up table and refined using CNNs, the next step is to estimate the pose of the flower in order for the end-effector to reach the center of each flower. This is an ill-posed problem due to the fact that the center of the flower could point toward any arbitrary position. This makes the task of estimating flower pose very challenging. We observed from the collected data that the orientation of flowers can be roughly put into three classes: the center points towards the center of the camera $c1$, towards the left of the camera $c2$, and towards the right of the camera $c3$ as shown in Figure 4.4. Thus, our pose estimation task is simplified to a multi-class classification problem, which can be solved using CNNs. Similar to the refinement method described in the segmentation step, we propose to address the pose estimation by retraining another network with three classes. The obtained flower pose information is then sent to help with adjusting the position of the end-effector.

(a) c1 (b) c2 (c) c3

Figure 4.4: Examples of orientation classes where the center of the flower is pointing at the center of the camera c_1, towards the left of the camera c_2 , and towards the right of the camera c_3.

Transfer Learning

In our approach, a transfer learning technique was adopted by taking advantage of the body of pre-trained classification network, which has rich features. The softmax layer was modified by retraining the network to perform classification on new image dataset. In general, the network computes the probability of each label $k \in \{1...K\}$:

$$p(k|x) = \frac{\exp(z_k)}{\sum_{i=1}^{K} \exp(z_i)} \tag{4.1}$$

where x is a training example, z_i is the logits or unnormalized log probability of each class [92], and k is either flowers or non-flowers in this context. The loss function is defined as

$$\ell = -\sum_{k=1}^{K} \log(p(k))q(k) \tag{4.2}$$

where $q(k)$ is the ground-truth distribution. The above cross entropy loss function is differentiable with respect to the log probability z_k which allows the use of gradient descent for training the neural networks. The gradient is bounded between -1 and 1 and has the following form:

$$\frac{\partial \ell}{\partial z_k} = p(k) - q(k). \tag{4.3}$$

Figure 4.5 shows some examples of classification applied to image patches extracted from the initial segmentation algorithm. The patches in the top row are classified as non-flower with probabilities 99.8%, 61.2%, and 61.2%, respectively. The patches in the bottom row are identified as flower with probabilities 91.1%, 97%, and 84.3%, respectively.

4.4 Model Selection

We picked four network architectures for transfer learning to experiment with, each of which has good performance on image recognition. They are **MobileNet V2** [84], **Inception V3** [92], **ResNet 152** [33], and **NasNet** [114]. They are described in the following.

4.4.1 ResNet

In order to address the problem of gradient descent in the training of deeper neural networks, ResNet [33] is proposed which has the residual learning structure. It is easier to optimize, and gains accuracy as network depths increases. It won 1st place on the ILSVRC 2015 image classification task with 152 layers. It also achieved 28% relative improvement on the COCO object detection dataset. There are some other titles ResNet has won such

Figure 4.5: Examples of classification applied to image patches extracted from the segmentation algorithm. The patches in the top row are classified as non-flower with probabilities 99.8%, 61.2%, and 61.2%, respectively. The patches in the bottom row are identified as flower with probabilities 91.1%, 97%, and 84.3%, respectively.

as 1st places on ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

The basic structure of a ResNet module is shown in Figure 4.6. It can be described using the following equation:

$$y = f(x) + x, \tag{4.4}$$

where x refers to the input layer, $f(x)$ stands for the output of each layer following the input layer, y refers final output. In deep learning, recovering the residual between input and output layer is called Deep Residual Learning (DRL). DRL has been widely used in many computer vision tasks such as image restoration and recognition.

4.4.2 Inception V3

Google's Inception V3 network has 42 layers with similar complexity as VGGNet. It achieved 1st runner up for image classification in ILSVRC 2015. It is trained for the Im-

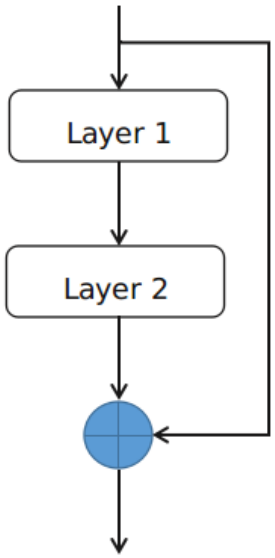


Figure 4.6: Basic structure of a ResNet module.

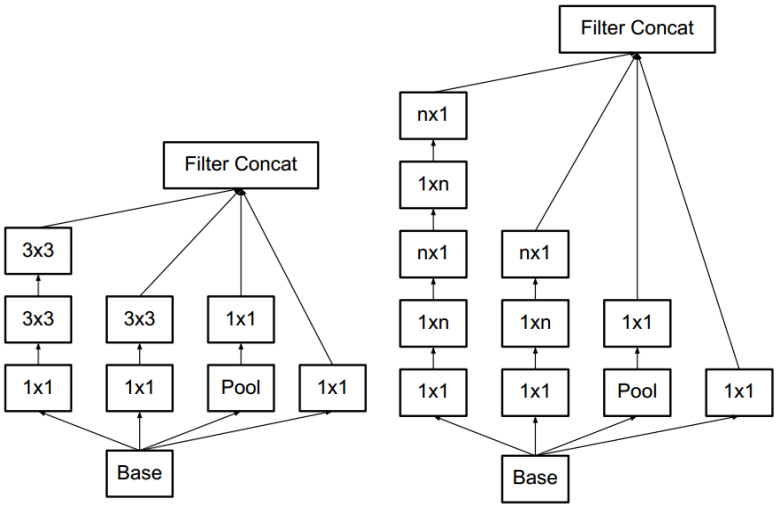
ageNet Large Visual Recognition Challenge dataset with 1000 classes. The Inception-v3 has some good feature extraction capabilities such as edge and shape detection.

Figure 4.7 shows three basic building blocks of the Inception V3 network. It features factorizing convolutions including both symmetrical one (Module A) and asymmetrical ones (Module B and C). The aim of using factorizing convolutions is to reduce the number of connections and parameter without decreasing the network efficiency. Some other techniques used in the network are auxiliary classifiers and efficient grid size reduction.

4.4.3 MobileNet V2

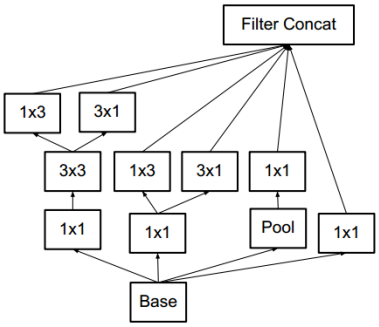
MobileNet [84] is a light-weight, low-latency, and small models to deal with constrained resources. It can be used to perform classification, detection, segmentation, etc. It is a trade-off between size and accuracy. Despite it's small size, it achieves competitive performance against other large networks such as Inception and ResNet on a ImageNet and COCO dataset.

The basic architecture of MobileNet is shown in Figure 4.8. It features an inverted residual block with depthwise convolution. Depthwise convolution is the key to make



(a) Inception module A

(b) Inception module B



(c) Inception module C

Figure 4.7: Three types of modules used in Inception V3 network. Module A uses factorization, module B and C uses asymmetric factorization.

MobileNet light-weight. Traditional convolutional layers uses large number of channels in previous layer as receptive fields, whereas depthwise convolution only use a certain number of channels as shown in Figure 4.8. As a result, the bottleneck layer has smaller size, compared with traditional residual learning blocks.

4.4.4 NasNet

NasNet [114] proposed a technique called NasNet search space which learns the model architecture on the dataset of interest. It searches the best architecture on a small scale dataset CIFAR-10 and then apply the architecture to ImageNet dataset with more copies of the cell. It achieved state-of-the-art accuracy of 82.7% top-1 and 96.2% top-5 on Ima-

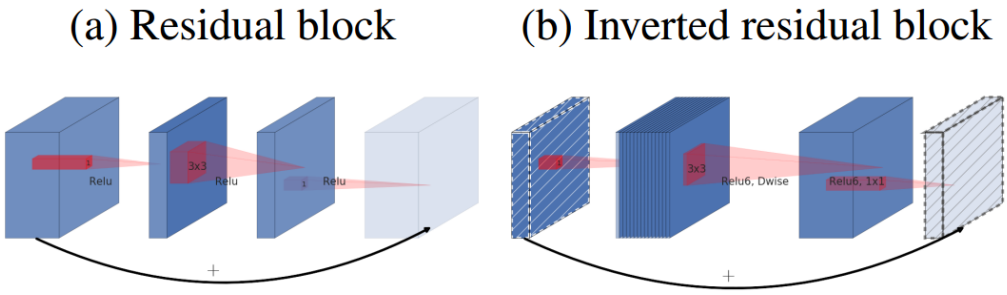


Figure 4.8: The inverted residual block architecture of MobileNet V2 [84].

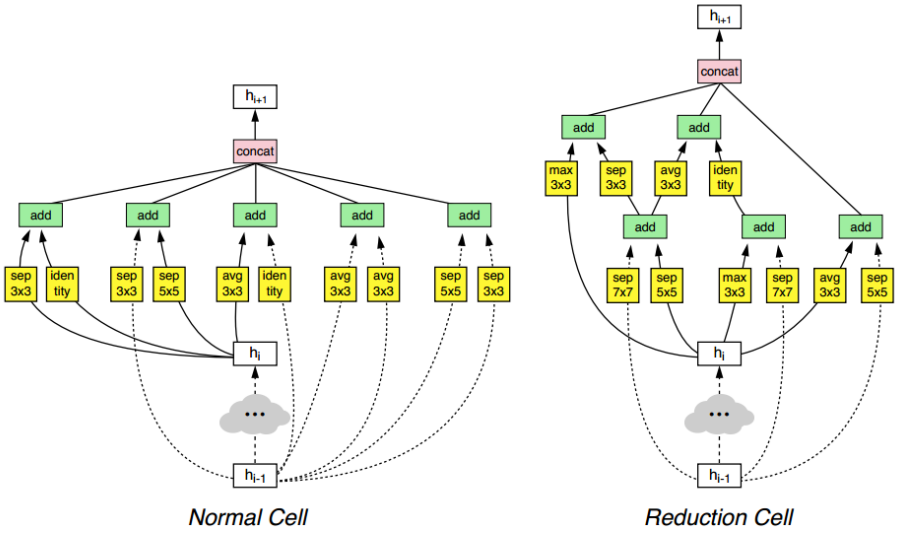


Figure 4.9: Architecture of NasNet [114] is composed of a normal layer (left) and reduction layer (right). Each cell includes 5 convolutional blocks.

geNet. It's basic building blocks including normal layer and reduction layer are shown in Figure 4.9.

Picking the right network for flower and pose classification requires significant empirical study. The main factor in picking the networks is the availability of training data. The current available training data is shown in Table 4.1. We have large amount of data for flower classification, whereas considerably less data for pose due to only a small number of patches show the full profile of a flower. Based on the dataset, we tried the aforementioned four networks with transfer learning, and the result is shown in Table 4.2. We found Inception and NasNet performs well toward flower classification, while we pick Inception

Table 4.1: The number of training patches for flower and pose classification.

	Class	Training
Flower	Pos	13,395
	Neg	14,105
Pose	C1	260
	C2	408
	C3	324

Table 4.2: Classification performance of each model on our dataset.

Model	Flower	Pose	Time (second)	Year	Input Size	Model Size
ResNet [33]	89.9%	67.4%	1.35	2016	224 × 224	233.8 MB
Inception [92]	90.3%	61.3%	0.58	2016	299 × 299	87.5 MB
MobileNet [84]	88.5%	76.3%	0.26	2018	224 × 224	9.2 MB
NasNet [114]	90.8%	64.8%	2.03	2018	331 × 331	340.6 MB

because it's less running time per image. With respect to pose, the result shows that small network such as MobileNet gives better performance due to the limited amount of training data. Thus, MobileNet is selected for pose classification.

4.5 Experiments and Results

4.5.1 Data Collection

In order to develop computer vision algorithms to help with robotics pollination, it is essential to collect sufficient amount of first-hand real flower data to test and validate the algorithms. We went to Bob's farm when the flowers are nearly and fully blooming. The camera we used is a Canon 5DS DSLR camera with a wide-angle fish eye lens. We set the time laps of the camera by taking one photo every 3 seconds, the camera is hold by the operator, and the operator moves parallel with the plant. We select multiple distances between the plant and the camera, such as close (1 meter), median (2 meters), and far away (3 meters). With the above mentioned setting, we have taken around 2400 images in four days

of May and June 2017 respectively. For each image in the database, we manually label all the flowers as either single or a cluster depending on whether there is overlapping among flowers. For a single flower, the pose and stage of a flower is specified by comparing with ground truth. For a flower cluster, the number of flowers in the cluster is saved. All the aforementioned tasks are assigned to each team member, and each person is responsible for labeling a small portion of the data. Finally, the manually labeled ground truth are merged and saved for future use.

4.5.2 Implementation Details

In order to train the network toward refining the segmentation, the positive and negative patches were obtained by comparing initial segmentation results against manually labeled images. There are 13,395 positive and 14,105 negative patches extracted in total from the labeled images for training. For testing, 2,102 and 2,124 patches are selected. Regarding pose estimation, we obtain the training patch of each class directly from the manually labeled dataset. The number of training patches for each class is 260 (c1), 408 (c2), and 324 (c3). There are 35, 30, and 33 testing patches for c1, c2, and c3. In the training phase, we use 80% of data for training, and 10% for validation and testing respectively. The training is performed using an Intel i9-7920x CPU and an NVIDIA Titan Xp GPU in TensorFlow.

4.5.3 Hyper-Parameters Tuning

After we select the Inception V3 for flower classification, and MobileNet V2 for pose classification, the networks are fine-tuned by adjusting multiple hyper-parameters such as iteration, data augmentation including random crop/scale/brightness, learning rate, and batch size. Following part will discuss the fine-tuning process with respect to each hyper-parameter.

Iterations

Iteration refers to the number of times running the training process. It is essential to select the appropriate iteration number to avoid both under fitting and over fitting. Figure 4.10 shows the accuracy and cross entropy versus training iterations for flower and pose classification. Depending on the amount of available training data, we run 20,000 iterations toward training the flower classification network, and 4,000 iterations toward training the pose classification network. We observe from the figure that the training accuracy keeps increasing and training cross entropy keeps decreasing. But it is not suggested to use training accuracy and training cross entropy (orange lines in the Figure) as indications to pick a good iteration number. The reason is that it may indicates over fitting. One should use validation accuracy and validation cross entropy instead (blue lines in the Figure). It is better to select an iteration number such that the validation accuracy is high and the cross entropy is low. We pick **16,500** and **1,600** as the final training iteration number to train Inception and MobileNet respectively.

Data Augmentation

It is good practice to augment the training data, since doing so could potentially improve the network performance. Some parameters one can adjust in TensorFlow are *random crop*, *random scale*, and *random brightness*. However, with these parameter turned on, the training took around 8 hours for MobileNet, and it could take several days to train Inception due to large amount of training data. So we recorded the effect of using data augmentation to MobileNet only, as shown in Table 4.3. We found that adding **random scale** gives the best performance 80.6% when compared to other settings.

Learning Rate

Learning rate is a parameter used in back propagation which controls the rate of gradient descent. Larger learning rate trains the network more faster, at the risk of dropping into a

Table 4.3: The effect of adding data augmentation to testing accuracy.

Data Augmentation	Percentage	Testing Accuracy
no augmentation	N/A	78.5%
random crop	5	79.6%
random scale	5	80.6%
random brightness	5	78.5%
random scale & crop	5	79.6%

Table 4.4: The effect of adjusting learning rate to testing accuracy.

Learning Rate	Flower	Pose
0.01	78.5%	90.8%
0.001	75.3%	89.7%
0.0001	69.9%	88.1%

zone that no longer modify network weights. Smaller learning rate may result in very long training time. We have tried 0.01, 0.001, and 0.0001 for both Inception and MobileNet. As shown in Table 4.4, **0.01** learning rate appears to give the best result for both flower and pose classification network.

Batch Size

Batch size refers to how many number of images/patches are fed into the network for training at each step. Ideally, one should use the whole dataset for each step, but that makes the training very time consuming. Using only one image/patch per step is noisy because the one sample may not be a good representation of the whole dataset. Thus, it is important to pick an appropriate batch size to achieve efficiency and avoid noise. We tried 50, 100, 150, and 200, finally we pick **100** as batch size as shown in Table 4.5.

Table 4.5: The effect of adjusting batch size to testing accuracy.

Batch Size	Flower	Pose
50	90.7%	75.3%
100	90.8%	78.5%
150	90.7%	78.5%
200	90.7%	77.4%

Table 4.6: Flower and pose classification results

	Class	Training	Testing	Precision	Recall
Flower	Pos	13,395	2,102	78.6%	90%
	Neg	14,105	2,124	88.5%	75.8%
Pose	C1	260	35	100%	94.3%
	C2	408	30	80%	76.7%
	C3	324	33	80%	90.9%

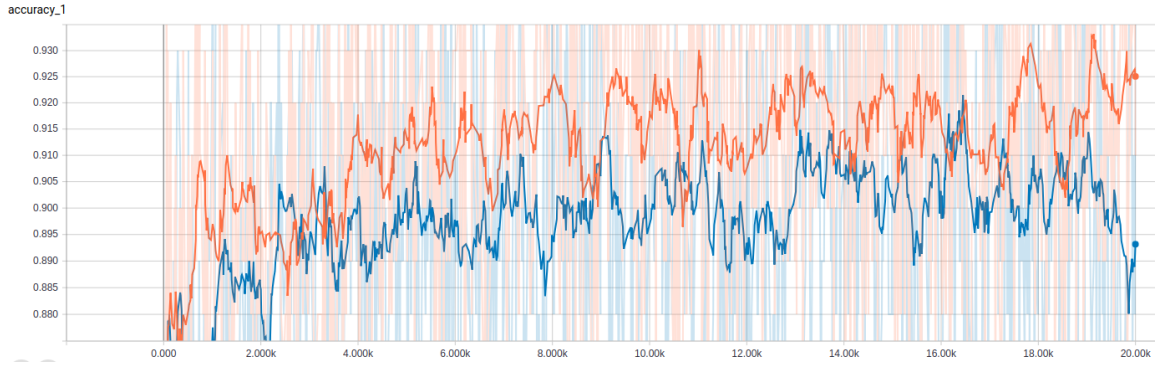
4.5.4 Results and Discussions

After the hyper-parameter tuning process, finally we pick 1600 and 16500 iterations for pose and flower, 5% random scale for pose, 0.01 learning rate, and 100 batch size. Under the aforementioned parameter setting, Table 4.6 shows the final flower and pose classification results with respect to precision and recall, as well as the number of patches used for training/testing. Our results show that the classifiers could reach as high as 90% and 94.3% recall for flower and pose classification respectively.

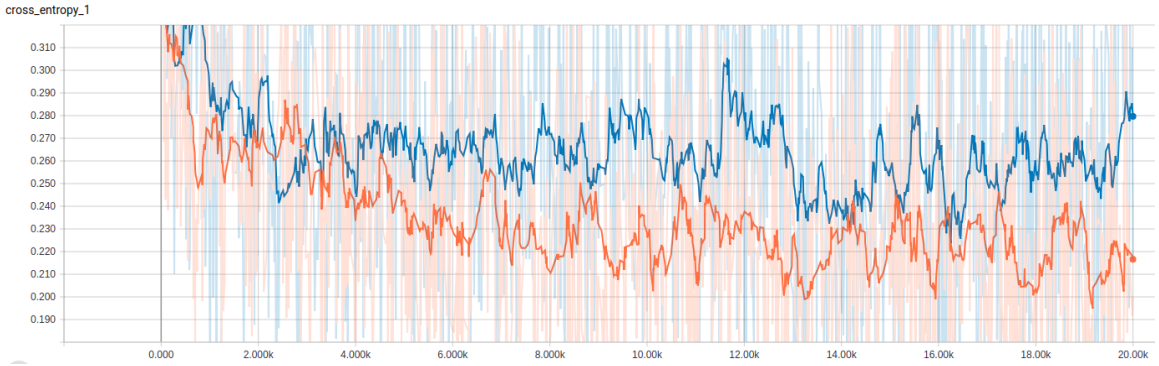
Figure 4.11 compares segmentation results using three examples. The first to the last column shows original image, manual labeling, SegNet [4] result, our result without flower classification, and with flower classification respectively. The numbers represent Jaccard similarity coefficient for image segmentation. The higher the coefficient indicates better performance. It is clear that flower segmentation is improved with the flower classification module.

4.6 Conclusion

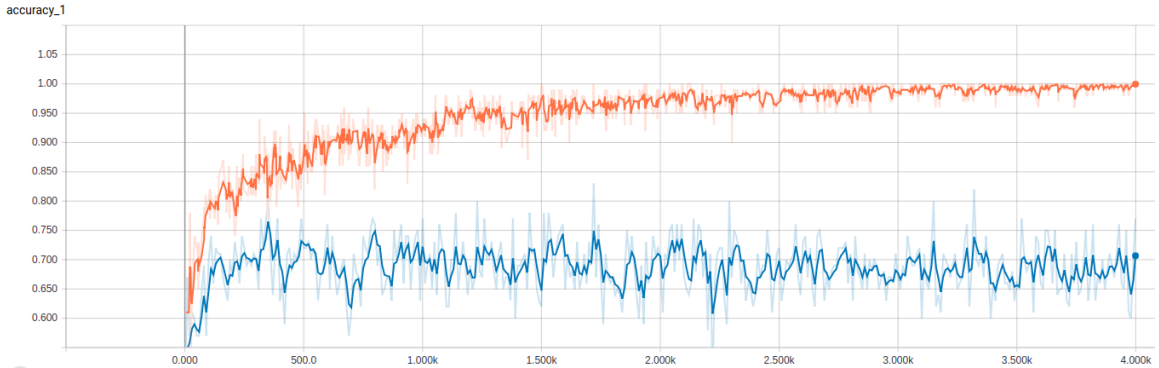
We describe a transfer learning-based approach assisting in robotic vision of a precision pollination robot named "BrambleBee". The robot is designed to fulfill the task of pollinating bramble plants. Our approach helps with flower and pose classification. Specifically, the flower classification module refines the initial flower segmentation by eliminating non-flower patches. The pose classification module estimates the pose of a flower from the pre-defined three pose classes. The flower and pose classifier are based Google Inception V3 and MobileNet V2, respectively. Large amount of flower data is collected from the farm and manually labeled. Good experimental results are reported based on real flower data.



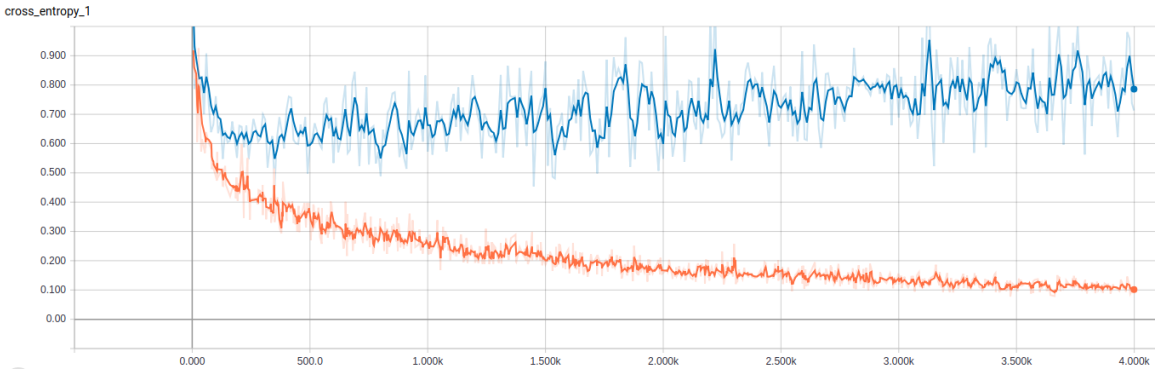
(a) Flower classification accuracy v.s. iterations.



(b) Flower classification cross entropy v.s. iterations.

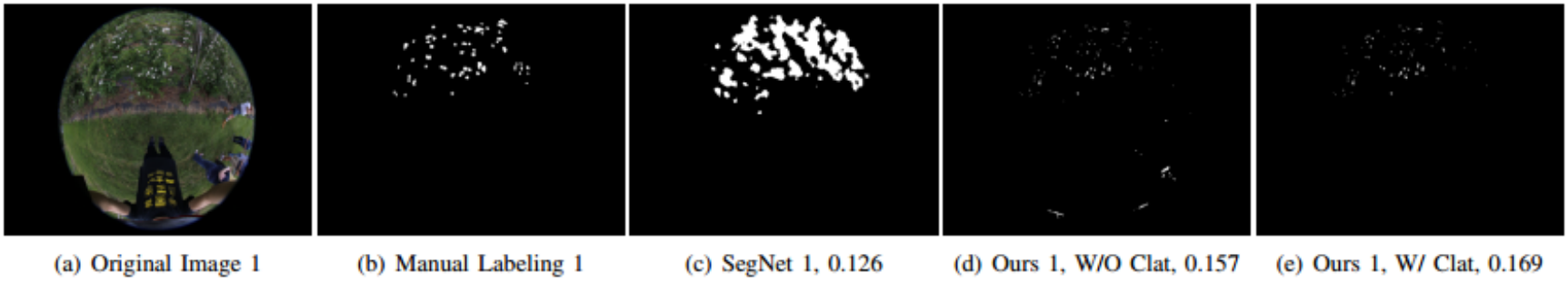


(c) Pose classification accuracy v.s. iterations.



(d) Pose classification cross entropy v.s. iterations.

Figure 4.10: The accuracy and cross entropy versus training iterations for flower and pose classification. Yellow line refers to training, and blue line refers to validation.



(a) Original Image 1 (b) Manual Labeling 1 (c) SegNet 1, 0.126 (d) Ours 1, W/O Clat, 0.157 (e) Ours 1, W/ Clat, 0.169

(f) Original Image 2 (g) Manual Labeling 2 (h) SegNet 2, 0.251 (i) Ours 2, W/O Clat, 0.167 (j) Ours 2, W/ Clat,, 0.223

(k) Original Image 3 (l) Manual Labeling 3 (m) SegNet 3, 0.152 (n) Ours 3, W/O Clat, 0.151 (o) Ours 3, W/ Clat, 0.283

Figure 4.11: Comparing segmentation results using three examples. The first to the last column shows original image, manual labeling, SegNet result, our result without flower classification, and with flower classification respectively. The numbers represent Jaccard similarity coefficient for image segmentation. The higher the coefficient indicates better performance.

Chapter 5

Conclusion

In this dissertation, we have studied deep learning for image restoration and robotic vision. We summarize the main contributions of this dissertation in the following.

In Chapter 2, we introduce a novel reformulation of the haze formation model. The reformulation allows us to solve the problem of single image dehazing using a deep residual network and a generative adversarial network. Specifically, our deep residual network focuses on recovering haze-free images from a distortion-minimization point of view. The generative adversarial network mainly emphasizes the perceptual visual quality of recovered images. We compare our result with several state-of-the-art approach in single image dehazing. Our approach has strong edge with respect to both distortion and perception.

In Chapter 3, we study the problem of incorporating deep learning into High Efficiency Video Coding (HEVC). Specifically, we built a loop filter module and a super-resolution module with deep learning. The loop filter module can be used to replace traditional loop filters in HEVC, and we show that our CNN-based loop filter module achieves around 7% BD-rate saving compared with baseline approach. The super-resolution module is designed to improve reconstructed image quality under low bandwidth. The proposed approach uses the down scaled input sequence by a factor of 2, then compress the sequence with a smaller quantization parameter (QP), and finally up scale the reconstruction back to the

original resolution using a multi-scale super-resolution CNN. We show experimentally that the super-resolution module is able to reduce distortion and improve visual quality when compared with baseline approach produced under the same bandwidth.

Chapter 4 explores implementing CNNs to help with autonomous robotic vision. We solve two image classification problems using the transfer-learning technique. This technique allows us to reuse the body of any pre-trained networks which contain rich edge and shape features, and only retrain the last softmax layer using the new dataset. With the extensive empirical study, we propose to use Google's Inception V3 network to address the flower classification problem, and use MobileNet V2 to deal with the pose classification problem. We have collected and manually labeled large amount of flower data from a local farm. We have shown that the proposed technique has good performance on the real flower data collected.

In the future, we plan to try some latest neural network architectures such as the Relativistic GAN [44] with respect to image dehazing. For image SR in HEVC, first we plan to study the impact of using multiple inputs with different scales as shown in Figure 3.23. Second, it is interesting to try compression of neural networks such as Deep Compression [29]. This technique could potentially reduce the network overhead that needs to be transmitted to the decoder. Regarding the robotic vision, the current image processing module has three separate blocks as shown in Figure 4.2, it is desirable to train an end-to-end network which takes a flower image as input, segment the flower patches and output the corresponding pose information.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2481–2495, 2016.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[6] D. Berman, S. Avidan, et al. Non-local image dehazing. In *Proceedings of the IEEE conference on CVPR*, pages 1674–1682, 2016.

[7] D. Berman, T. Treibitz, and S. Avidan. Air-light estimation using haze-lines. In *Computational Photography (ICCP), 2017 IEEE International Conference on*, pages 1–9. IEEE, 2017.

[8] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012.

[9] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012.

[10] Y. Blau and T. Michaeli. The perception-distortion tradeoff. In *Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, Utah, USA*, pages 6228–6237, 2018.

[11] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, 2016.

[12] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *null*, pages 275–282. IEEE, 2004.

[13] C. Chen, M. N. Do, and J. Wang. Robust image and video dehazing with visual artifact suppression via gradient residual minimization. In *ECCV*, pages 576–591. Springer, 2016.

[14] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1256–1272, 2017.

[15] L. Czuni, G. Csaszar, and A. Licsar. Estimating the optimal quantization parameter in h.264. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 330–333, Aug 2006.

[16] Y. Dai, D. Liu, and F. Wu. A convolutional neural network approach for post-processing in hevc intra coding. In *International Conference on Multimedia Modeling*, pages 28–39. Springer, 2017.

[17] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[18] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. Springer, 2014.

[19] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on PAMI*, 38(2):295–307, 2016.

[20] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016.

[21] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[22] R. T. et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1110–1121, July 2017.

[23] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics (TOG)*, 30(2):12, 2011.

[24] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International journal of computer vision*, 40(1):25–47, 2000.

[25] C.-M. Fu, E. Alshina, A. Alshin, Y.-W. Huang, C.-Y. Chen, C.-Y. Tsai, C.-W. Hsu, S.-M. Lei, J.-H. Park, and W.-J. Han. Sample adaptive offset in the hevc standard.

IEEE Transactions on Circuits and Systems for Video technology, 22(12):1755–1764, 2012.

[26] K. B. Gibson, D. T. Vo, and T. Q. Nguyen. An investigation of dehazing effects on image and video coding. *IEEE Transactions on Image Processing*, 21(2):662–673, 2012.

[27] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 349–356. IEEE, 2009.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[29] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[30] K. He, J. Sun, and X. Tang. Guided image filtering. In *European conference on computer vision*, pages 1–14. Springer, 2010.

[31] K. He, J. Sun, and X. Tang. Single image haze removal using dark channel prior. *IEEE transactions on PAMI*, 33(12):2341–2353, 2011.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on CVPR*, pages 770–778, 2016.

[34] X. He, Q. Hu, X. Zhang, C. Zhang, W. Lin, and X. Han. Enhancing hevc compressed videos with a partition-masked convolutional neural network. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 216–220. IEEE, 2018.

[35] Y. Hu, X. Gao, J. Li, Y. Huang, and H. Wang. Single image super-resolution via cascaded multi-scale cross network. *arXiv preprint arXiv:1802.08808*, 2018.

[36] J. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, June 2015.

[37] X. Huang, Y. Li, O. Poursaeed, J. E. Hopcroft, and S. J. Belongie. Stacked generative adversarial networks. In *CVPR*, volume 2, page 3, 2017.

[38] C. Hung, J. Underwood, J. Nieto, and S. Sukkarieh. A feature learning based approach for automated fruit yield estimation. In *Field and Service Robotics*, pages 485–498. Springer, 2015.

[39] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[40] M. Irani and S. Peleg. Improving resolution by image registration. 1991.

[41] K. Jia, X. Wang, and X. Tang. Image transformation based on learning dictionaries across image spaces. *IEEE transactions on pattern analysis and machine intelligence*, 35(2):367–380, 2013.

[42] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[43] D. A. Johnson, D. J. Naffin, J. S. Puhalla, J. Sanchez, and C. K. Wellington. Development and implementation of a team of robotic tractors for autonomous peat moss harvesting. *Journal of Field Robotics*, 26(6-7):549–571, 2009.

[44] A. Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[45] S. kak Kwon, A. Tamhankar, and K. Rao. Overview of h.264/mpeg-4 part 10. *Journal of Visual Communication and Image Representation*, 17(2):186 – 216, 2006. Introduction: Special Issue on emerging H.264/AVC video coding standard.

[46] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi. Residual interpolation for color image demosaicking. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 2304–2308. IEEE, 2013.

[47] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on CVPR*, pages 1646–1654, 2016.

[48] J. Kim, J. Kwon Lee, and K. Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016.

[49] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[51] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. *Proceedings of the IEEE conference on CVPR*, 2017.

[52] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[53] C. Lehnert, A. English, C. McCool, A. W. Tow, and T. Perez. Autonomous sweet pepper harvesting for protected cropping systems. *IEEE Robotics and Automation Letters*, 2(2):872–879, 2017.

[54] B. Li, X. Peng, Z. Wang, J. Xu, and D. Feng. Aod-net: All-in-one dehazing network. In *Proceedings of the IEEE Conference on CVPR*, pages 4770–4778, 2017.

[55] J. Li, F. Fang, K. Mei, and G. Zhang. Multi-scale residual network for image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 517–532, 2018.

[56] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao. Fully connected network-based intra prediction for image coding. *IEEE Transactions on Image Processing*, 27(7):3236–3247, 2018.

[57] R. Li, J. Pan, Z. Li, and J. Tang. Single image dehazing via conditional generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8202–8211, 2018.

[58] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 136–144, 2017.

[59] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2016.

[60] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[61] P. Lottes, M. Hörferlin, S. Sander, and C. Stachniss. Effective vision-based classification for separating sugar beets and weeds for precision farming. *Journal of Field Robotics*, 34(6):1160–1178, 2017.

[62] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss. Uav-based crop and weed classification for smart farming. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3024–3031. IEEE, 2017.

[63] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. 2:416–423 vol.2, July 2001.

[64] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, Oct 2017.

[65] G. Meng, Y. Wang, J. Duan, S. Xiang, and C. Pan. Efficient image dehazing with boundary constraint and contextual regularization. In *Proceedings of the IEEE ICCV*, pages 617–624, 2013.

[66] W. K. Middleton. Vision through the atmosphere. In *Geophysik II/Geophysics II*, pages 254–287. Springer, 1957.

[67] A. Milioto, P. Lottes, and C. Stachniss. Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2229–2235. IEEE, 2018.

[68] A. Mittal, A. K. Moorthy, and A. C. Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708, 2012.

[69] J. Moonrinta, S. Chaivivatrakul, M. N. Dailey, and M. Ekpanyapong. Fruit detection, tracking, and 3d reconstruction for crop mapping and yield estimation. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 1181–1186. IEEE, 2010.

[70] K. Nishino, L. Kratz, and S. Lombardi. Bayesian defogging. *International Journal of Computer Vision*, 98(3):263–278, 2012.

[71] S. Nuske, K. Wilshusen, S. Achar, L. Yoder, S. Narasimhan, and S. Singh. Automated visual yield estimation in vineyards. *Journal of Field Robotics*, 31(5):837–860, 2014.

[72] N. Ohi, K. Lassak, R. Watson, J. Strader, Y. Du, C. Yang, G. Hedrick, J. Nguyen, S. Harper, D. Reynolds, et al. Design of an autonomous precision pollination robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7711–7718. IEEE, 2018.

[73] M. Ollis and A. Stentz. Vision-based perception for an automated harvester. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS'97*, volume 3, pages 1838–1844. IEEE, 1997.

[74] J. Pang, O. C. Au, and Z. Guo. Improved single image dehazing using guided filter. *Proc. APSIPA ASC*, pages 1–4, 2011.

[75] W.-S. Park and M. Kim. Cnn-based in-loop filtering for coding efficiency improvement. In *Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016 IEEE 12th*, pages 1–5. IEEE, 2016.

[76] A. Paszke, S. Gross, S. Chintala, and G. Chanan. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, 2017.

[77] A. Payne, K. Walsh, P. Subedi, and D. Jarvis. Estimating mango crop yield using image analysis using fruit at stone hardening stage and night time imaging. *Computers and Electronics in Agriculture*, 100:160 – 167, 2014.

[78] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[79] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang. Single image dehazing via multi-scale convolutional neural networks. In *ECCV*, pages 154–169. Springer, 2016.

[80] W. Ren, L. Ma, J. Zhang, J. Pan, X. Cao, W. Liu, and M.-H. Yang. Gated fusion network for single image dehazing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3253–3261, 2018.

[81] I. E. Richardson. H. 264 and mpeg-4 video compression: video coding for next-generation multimedia. John Wiley & Sons, 2004.

[82] A. Ruckelshausen, P. Biber, M. Dorna, H. Gremmes, R. Klose, A. Linz, R. Rahe, R. Resch, M. Thiel, D. Trautz, and U. Weiss. Bonirob: An autonomous field robot platform for individual plant phenotyping. *Precision Agriculture*, 9:841–847, 01 2009.

[83] I. Sa, Z. Chen, M. Popović, R. Khanna, F. Liebisch, J. Nieto, and R. Siegwart. weednet: Dense semantic weed classification using multispectral images and mav for smart farming. *IEEE Robotics and Automation Letters*, 3(1):588–595, 2018.

[84] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[85] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I. IEEE, 2003.

[86] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. pages 3791–3799, June 2015.

[87] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient

Sub-Pixel Convolutional Neural Network. *arXiv e-prints*, page arXiv:1609.05158, Sep 2016.

[88] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. *ECCV*, pages 746–760, 2012.

[89] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[90] D. Slaughter, D. Giles, and D. Downey. Autonomous robotic weed control systems: A review. *Computers and electronics in agriculture*, 61(1):63–78, 2008.

[91] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[92] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[93] R. T. Tan. Visibility in bad weather from a single image. In *Proc. of CVPR*, pages 1–8. IEEE, 2008.

[94] J.-P. Tarel and N. Hautiere. Fast visibility restoration from a single color or gray level image. In *Proc. of ICCV*, pages 2201–2208. IEEE, 2009.

[95] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In D. Cremers, I. Reid, H. Saito, and M.-H. Yang, editors, *Computer Vision – ACCV 2014*, pages 111–126, Cham, 2015. Springer International Publishing.

[96] G. Valenzise, M. Tagliasacchi, and S. Tubaro. Estimating qp and motion vectors in h.264/avc video from decoded pixels. pages 89–92, 10 2010.

[97] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *The European Conference on Computer Vision Workshops (ECCVW)*, September 2018.

[98] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[99] Z. e. a. Wang. Deep networks for image super-resolution with sparse prior. 2015.

[100] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, pages 341–349, 2012.

[101] R. Xu, C. Li, and J. M. Velni. Development of an autonomous ground robot for field high throughput phenotyping. *IFAC-PapersOnLine*, 51(17):70 – 74, 2018. 6th IFAC Conference on Bio-Robotics BIOROBOTICS 2018.

[102] C.-Y. Yang and M.-H. Yang. Fast direct super-resolution by simple functions. In *Proceedings of the IEEE international conference on computer vision*, pages 561–568, 2013.

[103] e. a. Yang, Jianchao. Image super-resolution via sparse representation. 2010.

[104] J. Yang, Z. Lin, and S. Cohen. Fast image super-resolution based on in-place example regression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1059–1066, 2013.

[105] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *IEEE transactions on image processing*, 21(8):3467–3478, 2012.

[106] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. 2008.

[107] R. Yang, M. Xu, T. Liu, Z. Wang, and Z. Guan. Enhancing quality for hevc compressed videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[108] R. Zeyde, E. Michael, and P. Matan. On single image scale-up using sparse-representations. 2010.

[109] Y. Zhai and D. Ji. Single image dehazing for visibility improvement. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(1):355, 2015.

[110] H. Zhang and V. M. Patel. Densely connected pyramid dehazing network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[111] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 2017.

[112] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018.

[113] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2472–2481, 2018.

[114] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.